Fibonacci numbers and the Stern-Brocot tree in Coq

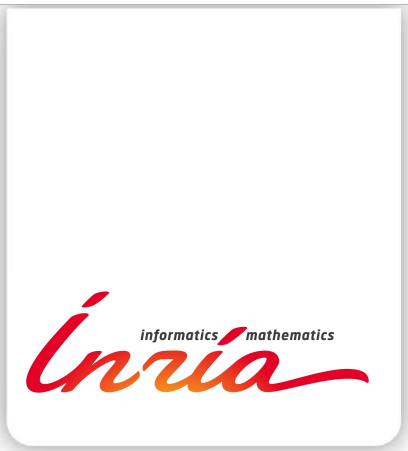
José Grimm

Fibonacci numbers and the Stern-Brocot tree in Coq

José Grimm

RESEARCH REPORT
N° 8654
December 2014
Project-Team Marelle

ISSN 0249-6399 ISRN INRIA/RR--8654--FR+ENG



Fibonacci numbers and the Stern-Brocot tree in Coq

José Grimm*

Project-Team Marelle

Research Report n° 8654 — December 2014 — 78 pages

Abstract: In this paper, we study the representation of a number by some other numbers. For instance, an integer may be represented uniquely as a sum of powers of two; if each power of two is allowed to appear at most twice, the number of representations is s_n, a sequence studied by Dijkstra, that has many nice properties proved here with the use of the proof assistant Coq. It happens that every rational number x is uniquely the quotient s_n/s_{n+1} as noticed by Stern, and that the integer n is related to the continued fraction expansion of x. It happens that by reverting the bits on n, one gets a sequence of rational numbers with increasing denominators that goes from 1 to x and becomes nearer at each iteration; this was studied by Brocot, whence the name Stern-Brocot tree. An integer can also be represented as a sum of Fibonacci numbers; we study R(n) the number of such representations; there is uniqueness for the predecessors of Fibonacci numbers; there is also uniqueness under additional constraints (for instance, no two consecutive Fibonacci numbers can be used, or no two consecutive numbers can be omitted). The code is available on the Web, under http://www-sop.inria.fr/marelle/gaia.

Key-words: Coq, SSreflect, Stern Brocot, Fibonacci, sequences, representation

* Email: Jose.Grimm@inria.fr

RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Les nombres de Fibonacci et l'arbre de Stern Brocot dans Coq

Résumé : Dans ce papier nous étudions les représentations d'un nombre au moyen d'autres nombres. Par exemple un entier peut être représenté de façon unique par une somme de puissances de deux, si chaque puissance de deux peut être utilisée au plus deux fois, le nombre de possibilités est s_n, une suite étudiée par Dijkstra, et qui possède de nombreuses propriétés intéressantes, prouvées formellement avec l'assistant de preuve Coq. Il s'avère que tout nombre rationnel s'écrit manière unique comme quotient s_n/s_{n+1}, ansi que l'a remarqué Stern; l'entier n est relié au développement en fraction continue du rational x. Si l'on prend la représentation binaire de n dans le sens opposé, on obtient une suite de nombres rationnels allant de 1 à x, en approchant de mieux en mieux x; ceci a été étudié par Brocot, d'où le nom d'arbre de Stern-Brocot. Un entier peut également être représenté comme somme de nombres de Fibonacci; nous étudions les propriétés de $\mathrm{R}(n)$, le nombre de telles décompositions. Il y a unicité pour les prédécesseurs des nombre de Fibonacci; il y a également unicité si l'on rajoute des contraintes (il y a unicité deux nombres de Fibonacci consécutifs sont exclus, ou si la distance entre deux nombres est au maximum deux).

Le code est disponible sur le site Web http://www-sop.inria.fr/marelle/gaia.

Mots-clés : Coq, SSreflect, Stern Brocot, Fibonacci, séquences, représentation

Chapter 1

Introduction

In this paper, we study the properties of some sequences of integers. An integer will be an element of the data type `nat`, and a sequence has type '`nat -> nat`'. We shall also consider finite sequences, of type '`seq nat`'. Any non-zero integer n can be represented in base b by a sequence $(a_i)_i$ of type '`seq 'I_`b' via the relation $n = \sum_i a_i b^i$. Uniqueness happens when the last element of the sequence is non-zero. The type of the sequence ensures $a_i < b$. In particular, if $b = 2$, each a_i is zero or one, the last a_i is 1 and may be omitted. This gives an efficient representation of non-zero integers (the data type `positive`), thus of all integers (the data type `N`).

We shall consider what happens, for the base two, when $a_i < b$ is replaced by $a_i \leq b$; how many possibilities are there? this means to compute the cardinal of a subset X of $\mathbf{N} \to \mathbf{N}$ (or of a subset of the set of finite lists over $\mathbf{N}$) which is not obvious, since the SSREFLECT library defines cardinals only for subsets of a finite set Y. We study a similar question: given an infinite sequence b_i, is it possible to represent every integer as $\sum_i a_i b_i$, where a_i is zero or one, in other terms, is it possible to represent the number in the form $\sum_{i \in \mathrm{I}} b_i$ where I is some finite set? If so, do we have uniqueness? how many possibilities are there? are there simple conditions on I in order to get uniqueness? We shall consider the Fibonacci sequence as a example.

It is well-known that the sets $\mathbf{N}$ and $\mathbf{Q}$ are countable, thus have the same cardinal. Finding an explicit bijection is not easy. One could proceed as follows: let $f(a,b)$ be $(a+1,b)$ if $a+1 < b$ and $(1,b+1)$ otherwise; let $f'(a,b)$ be $f^k(a,b)$, where $k > 0$ is the least integer such that the two components of the k-th iteration of f are coprime. Let g be defined by induction by $g(0) = 1/2$, and if $g(n) = a/b$, $f'(a,b) = (a',b')$ then $g(n+1) = a'/b'$. This function is injective, and the range is the set of all rational numbers x such that $0 < x < 1$. It is easy to deduce a bijection $\mathbf{N} \to \mathbf{Q}$. There is however a simple isomorphism $\mathbf{N} \to \mathbf{Q}^+$, associated to the Stern-Brocot (or Calkin-Wilf) tree. We shall study the properties of the sequence associated to this tree, as well as the properties of the Stern diatomic sequence, and the relationship with continued fractions. Note that rational numbers (the data type `rat`) are built upon the signed integers $\mathbf{Z}$ (the data type `int`).

In this chapter we consider some lemmas that will be useful for the sequel. We shall also define the Fibonacci numbers, as they are not in the SSREFLECT library. The code is available on the Web at http://www-sop.inria.fr/marelle/gaia. The file *fibm.v* contains the code described in the first two chapters, and *stern.v* the code of the two other chapters.

1.1 Additional lemmas

We will be faced with a problem of notations: for instance $a+b$ means addition; we could use explicit names like `addn`, `addz` or `addq`, but we prefer use the scoping mechanism. The default interpretation will be in `nat_scope`, and `(x)%nat` forces x to be interpreted in this scope; the SSREFLECT library rebinds `(x)%N` (interpretation as a positive number) as `(x)%num` and uses `(x)%N` as short for `(x)%nat`. Sometimes the default scope will be `ring_scope`, and `(x)%R` forces x to be interpreted in this scope. Note that `(x)%:Q` means x coerced into **Q**, while `(x)%:Z` means x coerced into **Z**.

The notations `n.+1`, `n.-1`, `n.*2`, and `n./2` refer to the successor of n, its predecessor, its double, its half, when n is a natural number. There is no notation for the square. On any ring, `x*+2` and `x^+2` denote the double or square of x (and 2 can be replaced by any natural number).

We consider here some useful small lemmas such as $(m+1)/2 \leq m$ (on **Q** this relation holds only when $1 \leq m$, on **N**, the half of one is zero).

1.1.1 Functions on nat

Comparison.

```
Lemma ltn_paddl a b: 0 < a -> b < a + b.
Lemma ltn_paddr a b: 0 < a -> b < b +a.
Lemma leq_BD n p : n - p <= n + p.
Lemma leqn1 n: (n <=1) = (n==0) || (n ==1).
```

Half.

```
Lemma half_le1 m: m./2 <= m.
Lemma half_le2 m: (m.+1)./2 <= m.
Lemma half_le3 m: 1 < m ->  (m.+1)./2 < m.
Lemma half_le3w m: 1 < m ->  m./2 < m.
Lemma half_le4 n: n <= n./2 -> n <= 1.
Lemma double_half_le m : (m./2.*2 <= m).
```

Double.

```
Lemma double_le1 n: n <= n.*2.
Lemma double_le2 n: n < n.*2.+1.
Lemma double_le3 n: n.+1 < (n.+1)*2.
Lemma eqn_double m n: (m.*2  == n.*2) = (m == n).
Lemma doubleS_inj: injective (fun z : nat => z.*2.+1).
Lemma double_inj: injective (fun z : nat => z.*2).
```

Odd.

```
Lemma odd_dichot n:  n = (n./2).*2.+1 \/ n = (n./2).*2.
Lemma oddE n: odd n -> n = (n./2).*2.+1.
Lemma evenE n: odd n = false -> n = (n./2).*2.
```

Some other lemmas. The first says that two to the successor of n is twice two to the n. The next ones say $\mathrm{R}(f(m),2^{i+1}) = f(\mathrm{R}(m,2^i))$, where $\mathrm{R}(a,b)$ is the remainder in the division of a by b and $f(m)$ is $2m$ or $2m+1$. Finally we have $2^n = ((2^n-k)/3)\cdot 3+k$, where k is 1 if n is even and 2 otherwise (note that $k = \mathrm{R}(2^n,3)$, proof by induction).

```
Lemma cantor n : n < 2 ^n.
Lemma expn2S n: 2^n.+1 = (2^n).*2.
Lemma rem_two_prop1 m i: (m.*2) %% (2 ^(i.+1)) = (m %% 2^i).*2.
Lemma rem_two_prop2 m i: (m.*2.+1) %% (2 ^(i.+1)) = (m %% 2^i).*2.+1.
Lemma pow2_mod3 n: 2^n %% 3 = 1 + odd n.
Lemma pow2_mod3' n:
  2^n = if (odd n) then (3*(((2^n).-2) %/3)).+2 else (3*(((2^n).-1) %/3)).+1.
Lemma sqrnD_sub' m n: n <= m -> (m + n) ^ 2 =  4 * (m * n) + (m - n) ^ 2.
```

1.1.2 Functions on sums

We show here

$$\sum_{i<n} \mathrm{F}_i = \sum_{i<(n+1)/2} \mathrm{F}_{2i} + \sum_{i<n/2} \mathrm{F}_{2i+1}. \tag{1.1}$$

and som other small results (the codomain of F could be any abelian monoid)

```
Lemma split_sum_even_odd (F: nat -> nat) n:
  \sum_(i<n) (F i) = \sum_(i< (n.+1)./2) (F i.*2) + \sum_(i< n./2) (F i.*2.+1).
Lemma split_sum_even_odd1 (F: nat -> nat) n:
  \sum_(i<n.*2) (F i) = \sum_(i< n) (F i.*2) + \sum_(i< n) (F i.*2.+1).
Lemma sum_rcons a (l: seq nat) F:
  \sum_(i <- rcons l a) F i =  F a + \sum_(i <- l) F i.
Lemma sum_rev (l: seq nat)(F: nat -> nat):
  \sum_(i <- rev l) F i = \sum_(i <- l) F i.
```

We show here (by induction on $b-a$) that

$$\sum_{a+c\le i<b+c \mid p(i)} \mathrm{F}(i) = \sum_{a\le i<b \mid p(i+c)} \mathrm{F}(i+c).$$

() Thus sums $\sum_{a\le i<b|p(i)} \mathrm{F}_i$ with the same bounds and same value and value, but different predicates, are equal when the predicates have the same value for every integer in the bounds.

```
Lemma big_nat_shift  a b c p F:
  \sum_(c + a <= i < c + b | p i) F i =
  \sum_(a <= i < b | p (c + i)) F (c + i).
Lemma big_nat_cond_eq  a b p p' F :
  (forall i, a <= i < b ->  (p i = p' i)) ->
  \sum_(a <= i < b | p i) F i =  \sum_(a <= i < b | p' i) F i.
```

1.1.3 The totient function

Let's denote by $a \perp b$ the property that a and b are coprime. This is equivalent to say that there is a Bezout relation $ua = bv+1$. only when a is non-zero).

```
Lemma coprime_if_bezout a b u v: u * a  = v * b + 1 -> coprime a b.
Lemma minimal_bezout a b: 0 < a -> 0 < b -> coprime a b ->
  exists u v, [/\ u * a  = v * b + 1,  v < a & u <= b].
Lemma minimal_bezout_prop a b u v: u * a  = v * b + 1 -> v < a -> u <= b ->
  (forall u' v', u' * a = v' * b +1 -> (u <= u') && (v <= v')).
```

Let $\phi(n)$ be the cardinal of F_n, the multiplicative group modulo n. This is the number of integers k such that $k < n$ and $k \perp n$ (k is coprime to n). If $1 < n$ then $0 \notin F_n$ so that $\phi(n) < n$. The SSREFLECT library provides the formula $\phi(n) = \prod_p (p-1)p^{k-1}$, where the product is over all prime divisors of n and k is the exponent. In particular

$$\phi(p^k) = (p-1)p^{k-1} = p^k - p^{k-1} \qquad (k > 0, p \text{ prime}). \tag{1.2}$$

For $k = 1$, we get: if p is prime, then $\phi(p) = p-1$. The converse holds (if p is not prime, there is a divisor d such that $1 < d < p$, thus $d \notin F_p$ and $\phi(n) < p-1$).

Alternate proof. Let $n > 1$ be any integer. There is at least a prime p dividing n. Write $n = m \cdot p^k$ where $m \perp p^k$ (in particular $k > 0$). If $m = 1$, then $k > 1$, and equation (1.2) shows, after some computations, that $\phi(n) < n-1$. Relation

$$a \perp b \implies \phi(ab) = \phi(a)\phi(b). \tag{1.3}$$

together with $\phi(a) < a$ and $\phi(b) < b$ gives $\phi(a)+\phi(b)+\phi(ab) \leq ab-1$. The conclusion follows for $m > 1$ with $a = m$, $b = p^k$ since $\phi(a) > 0$.

```
Lemma totient_pfactor_alt p e: prime p -> 0 < e ->
    totient (p ^ e) = p ^ e - p ^ e.-1.
Lemma totient_ltn n: 1 < n -> totient n < n.
Lemma totient_prime n: prime n = (0 < n) && (totient n == n.-1).
```

1.1.4 Pythagorean triples

We solve here the equation

$$x^2 + y^2 = z^2. \tag{1.4}$$

We first state three auxiliary results. If x and y are two integers, g is their gcd, $x' = x/g$ and $y' = y/g$ then: $x = x'g$, $y = y'g$ and $x' \perp y'$. If $ab^2 = c^2$, then there is$d$ such that $c = bd$ and $a = d^2$ (note that b divides c). Finally, if $a \perp b$ and ab is a square then both a and b are squares, and their square roots are coprime (let g be the gcd of a and c, a', c' the cofactors so that $a'b = c'^2 g$. One deduces that a' divides g, being coprime to c'^2 and that g divides a', being coprime to b. So $a' = g$ and $a = g^2$).

There is no trivial solution of (1.4) with $x = y$. This means that $\sqrt{2}$ is an irrational number. Proof by induction. If $2y^2 = z^2$, then z is even, and $y^2 = 2(z/2)^2$. This gives a smaller solution.

```
Lemma gcd_aux x y: 0 < x -> exists x' y',
  [/\ x = x' * gcdn x y, y = y' * gcdn x y & coprime x' y'].
Lemma pythagore_aux a b c: 0 < b -> a * b^2 = c^2 ->
    exists d, c = d * b /\ a = d ^2.
Lemma factor_square a b c: coprime a b -> a * b = c ^2 ->
  exists u v, a = u ^2 /\ b = v^2.

Lemma double_square_square m n: (n ^2).*2 = m ^2 -> n = 0.
Lemma gcd_n2 n: gcdn n 2 = if (odd n) then 1 else 2.
Lemma square_odd_mod4 n: odd n -> n^2 = 1 %[mod 4].
```

Given three numbers p, q and r with $q \leq p$, we say that (x, y, z) is a *pythagorean triple* when $x = r(p^2 - q^2)$, $y = 2rpq$, $z = r(p^2 + q^2)$. This is a solution to (1.4). Example $p = 3$, $q = 1$

and $r = 1$ gives $x = 8$, $y = 6$, and $z = 10$. The same solution can be obtained, exchanging x and y, with $p = 2$, $q = 1$ and $r = 2$. The reason is that, if x and y are coprime, then $r = 1$ and $p \perp q$; the converse being false: it may happen that x and y are both even. Now, x is even only when p and q have the same parity, so that p and q are odd. So, when p and q have different parities, then x and y are coprime. Conversely, if x and y are coprime, $x^2 + y^2 = z^2$, then x and y have different parities.

```
Lemma pythagore_tripleA p q r
  (x := r* (p^2 - q^2)) (y := r*(p*q).*2) (z:= r*(p^2 + q ^2)):
  q <= p -> x^2 + y^2 = z ^2.

Lemma pythagore_gcd p q:  q <= p -> coprime p q ->  odd p != odd q ->
  coprime (p ^ 2 - q ^ 2) (p * q).*2.
Lemma pythagore_mod4 x y z: coprime x y -> x ^ 2 + y ^ 2 = z ^ 2 ->
  odd x != odd y.
Lemma coprime_sqr m n:  coprime (m ^2) (n ^ 2) = coprime m n.
Lemma coprimeDl m n: coprime m (m + n) = coprime m n.
Lemma coprimeDr m n: coprime m (n + m) = coprime m n.
```

We show here that the solutions of (1.4) are: either $x = 0$, case where $y = z$ is arbitrary, or $y = 0$, case where $x = z$ is arbitrary, or one of (x, y, z), (y, x, z) is a pythagorean triple associated to (p, q, r); in this case it can be assumed that $p < q$, p and q are coprime, exactly one of p, q is odd. Proof: if g is the gcd of x and y, then g^2 divides x^2 and y^2, thus divides z^2. This means that g divides z. After division by g, x and y become coprime; moreover z is coprime with x and y. By symmetry, we may assume x odd and y even, so that z is odd. Now $z - x$ and $z + x$ are even, and the equation becomes $y^2 = z^2 - x^2$ thus $(y/2)^2 = (z - x)/2(z + x)/2$. Since x and z are coprime it follows that the two factors are coprime, thus are squares, say q^2 and p^2. The quantities p and q solve the problem.

```
Lemma pythagore_tripleB x y z (r := gcdn x y): x^2 + y ^2 = z^2 ->
  [\/ x=0 /\ y = z, y = 0 /\ x = z |
  exists p q, [/\ 0 < r, q < p, coprime p q, odd p != odd q &
     [/\ x = r* (p^2 - q^2), y = r*(2*p*q) & z = r*(p^2 + q ^2) ] \/
     [/\ y = r* (p^2 - q^2), x = r*(2*p*q) & z = r*(p^2 + q ^2) ]]].
```

Application; $x^4 + y^4 = z^4$ has only trivial solutions. We prove in fact a stronger result: that $x^2 + y^4 = z^4$ has only trivial solutions. Proof by induction: given a solution, we exhibit another one with smaller z. First, if r is the gcd of y and z, then r^4 divides x^2 so that r^2 divides x so that we may assume $y \perp z$. This implies $x \perp y^2$. Assume first y odd. There are p and q such that $y^2 = p^2 - q^2$, $z^2 = p^2 + q^2$ and $x = 2pq$. It follows $(xy)^2 + q^4 = p^4$, absurd by induction. Assume now y even, so that x is odd. Rewrite the equation as $x^2 = (z^2 - y^2)(z^2 + y^2)$. Both factors are odd, thus are coprime, thus are squares. Write $z^2 - y^2 = t^2$ and $z^2 + y^2 = s^2$. Note that s and t are odd, and write $u = (s + t)/2$, $v = (s - t)/2$, so that $s = u + v$ and $t = u - v$. Note that $u^2 + v^2 = z^2$, $2uv = y^2$, $u \perp v$. So, there are p and q such that $u = p^2 - q^2$, $v = 2pq$, $z = p^2 + q^2$ (the roles of p and q may be exchanged). The conclusion follows by induction since $u = p^2 - q^2$, and the three quantities u, p and q are squares (note that $2uv$ is a square, so that u and $2v$ are squares; now pq is a square).

Alternate proof. Let $\mathrm{H}(x)$ be: there is an integer t such that $x = t^2$ or $x = 2t^2$. We show, by induction, that if $x^2 + y^2 = z^2$, $x \perp y$, then at most one of x, y and z can satisfy H. We may assume $x = 2pq$, $y = p^2 - q^2$ and $z = p^2 + q^2$. Since y and z are odd, $\mathrm{H}(y)$ and $\mathrm{H}(z)$ means that y and z are squares, as well as yz, and the conclusion follows from $q^4 = yz + p^4$. Assume

H(x). From $x = 2pq$ it follows that p and q satisfy H. If H(y) or H(z) holds, the conclusion follows from $z = p^2 + q^2$ or $q^2 + y = p^2$ respectively.

```
Lemma Fermat4 x y z : x^2 + y^4 = z^4 -> (x == 0) || (y == 0).
Lemma Fermat4_alt x y z (pa:= fun x => exists y, x= y^2 \/ x = (y^2).*2)
  (pb := fun x y z => [\/ pa x /\ pa y, pa y /\ pa z | pa z /\ pa x]):
  x^2 + y ^2 = z ^2 -> coprime x y -> pb x y z -> (x == 0) || (y == 0).
Lemma Fermat4' x y z : x^2 + y^4 = z^4 -> (x == 0) || (y == 0).
```

Example. Let's consider $x^2 + a = z^2$ for small a. If $x \geq a/2$, the only possible solution is $z = x + 1$. In particular, for $a = 1$, the only solution is $x = 0$ and $z = 1$, if $a = 2$ there is no solution, and if $a = 4$, the only solution is $x = 2$ and $z = 2$. There is an alternate proof in the case $a = 4$; we have either $2pqr = 2$, so that $p = q = 1$ and $y = 0$, or $2 = r(p^2 - q^2) = r(p - q)(p + q)$. The quantity $p + q$ divides 2, so must be 1 or 2; but $p + q$ is odd, and $q < p$; so $q = 0$ and $x = 0$. The case $a = 9$ is similar, but the subcase $p + q = 3$, $p - q = 1$ has a solution and we get $3^2 + 4^2 = 5^2$. The case $a = 16$ is similar: since $p - q$ and $p + q$ are odd, the relation $4 = r(p - q)(p + q)$ says $r = 4$. A direct proof is however shorter.

```
Lemma square_plus1_square m n: n ^2 + 1 = m ^2 -> n = 0.
Lemma square_plus2_square m n: n ^2 + 2 = m ^2 -> False.
Lemma square_plus4_square m n: n ^2 + 4 = m ^2 -> n = 0.
Lemma square_plus9_square n m: (n ^2 + 3^2 = m ^2) -> (n = 0 \/ n = 4).
Lemma square_plus16_square n m:
  (n ^2 + 4^2 == m ^2) = (n==0) && (m==4)||(n==3)&&(m==5).
```

1.1.5 Functions on lists

We consider here some properties of lists (called `seq` in SSREFLECT). A list is either empty (`nil`, denoted `[::]`) or formed by adding an element a in front of a list l, the operator is `cons`, the operation is denoted $[a :: l]$. In order to prove a property P on a list it suffices to prove it for the empty list, and to prove that, whatever a and l, P(l) implies P($a :: l$). We give a variant with two lists of the same size.

```
Lemma seq2_ind (T1 T2 : Type) (P : seq T1 -> seq T2 -> Prop) : P [::] [::] ->
  (forall x1 x2 s1 s2, P s1 s2 -> P (x1 :: s1) (x2 :: s2)) ->
  forall s1 s2, size s1 == size s2 -> P s1 s2.
```

Each non-empty list has a head and a tail; the `head` and `last` functions take a second argument, to be used when the list is empty; these functions will be denoted H(l) and T(l) (we shall omit the other argument). One can remove the head or tail of list, the operations are called `behead` and `belast`. For some reason, the latter function takes two arguments. The operations will be denoted by H$'$(l) and T$'$(l). One can revert a list, this operation is denoted by $r(l)$. We have, for instance, H($r(l)$) = T(l). One can remove an element from a list (note that only the first instance of the element is removed).

```
Lemma rev_inj  (T: Type) : injective (@rev T).
Lemma head_rev (T: Type) (a:T) l: head a (rev l) = last a l.
Lemma split_rev (T: Type) (a:T) l:
  rev (a :: l) = last a l :: behead (rev (a :: l)).
Lemma all_rev (T:eqType) P (l: seq T) : all P (rev l) = all P l.
Lemma rem_rcons1 a b l: a < b -> rem a (rcons l b) = rcons (rem a l) b.
Lemma rem_rcons2 (T:eqType) (a: T) l: a \notin l -> rem a (rcons l a) = l.
```

The notation `[f i | i <- l]` refers to the list obtained from l by apply f to each element; we shall be interested in the case where f is the successor or predecessor function. The construction '`mkseq f n`' produces the list $[f(0), f(1), \ldots]$ of length n; it is by definition '`[seq f i | i <- iota 0 n]`'. The function `iota` proceeds by adding a new element on the left; it may be viewed the other way around.

```
Definition succ_seq l :=  [seq i.+1 | i <- l].
Definition pred_seq l :=  [seq i.-1 | i <- l].
Lemma seq_prednK l: all (leq 1) l ->  l = succ_seq (pred_seq l).

Lemma iota_S a n: iota a.+1 n = succ_seq (iota a n).
Lemma iota_Sr i n: iota i n.+1 = rcons (iota i n)  (i+n).
Lemma last_iota a b c: last c (iota a b.+1) = (b + a).

Lemma mkseq_succ (T: Type) (f: nat -> T) n:
  mkseq f n.+1 = rcons (mkseq f n) (f n).
Lemma last_mkseq (f: nat -> nat) a b c:
  last c [seq f i | i <- iota a b.+1] = f (b + a).
```

1.1.6 Expansion to base two

A `positive` number is either x_{H}, $x_{\mathrm{O}}(k)$ or $x_{\mathrm{I}}(k)$ where k is positive; this can be interpreted as a natural number, as respectively 1, $2n$ or $2n+1$ if k is interpreted as n. Note that the interpretation is never zero. An element of `N` is either '`N0`' or '`Npos p`', where p is positive. One can interpret an element of `N` as a natural number.

Let c' the function that interprets an element of `N` as a natural number. This is called `N.to_nat` in the standard library. It is defined indirectly via an iterator I that takes two arguments, a value a and a binary function f. We have $\mathrm{I}(x_{\mathrm{H}}) = a$, $\mathrm{I}(x_{\mathrm{O}}(p), a) = \mathrm{I}(p, f(a,a))$ and $\mathrm{I}(x_{\mathrm{I}}(p), a) = f(a, \mathrm{I}(p, f(a,a)))$. This is applied to the case where $a = 1$ and f is addition. At the i-th iteration, the argument becomes 2^i. We prefer the SSREFLECT version `nat_of_bin`, which uses Horner Scheme (in the case of $x_{\mathrm{O}}(x_{\mathrm{I}}(x_{\mathrm{O}}(x_{\mathrm{I}}(x_{\mathrm{H}}))))$ it computes 1, 3, 6, 13, 27, 54, rather than all powers of two and the sum of 2, 4, 16 and 32.

Let c be the function that converts a natural number into an object of type `N`. This is called `N.of_nat` in the standard library, defined by $c(n+1) = s(c(n)$, where the successor function s on `N` is defined by $s(x_{\mathrm{O}}(k)) = x_{\mathrm{I}}(k)$ and $s(x_{\mathrm{I}}(k)) = x_{\mathrm{O}}(s(k))$. We prefer the SSREFLECT version `bin_of_nat`. It uses an auxiliary function that essentially inlines the division by two. The following lemmas show how the algorithm works.

```
Definition pos_of_nat' n := (pos_of_nat n n).
Lemma bin_of_nato n: bin_of_nat ((n.+1).*2.+1) = Npos (xI (pos_of_nat' n).
Lemma bin_of_nate n: bin_of_nat (n.+1).*2 = Npos (xO (pos_of_nat' n).

Lemma nat_of_pos_ne0 p: nat_of_pos p != 0.
Lemma pos_of_natK n: nat_of_pos (pos_of_nat' n) = n.+1.
Lemma nat_of_posK p: pos_of_nat' (nat_of_pos p).-1 = p.
Lemma nat_of_pos_inj: injective nat_of_pos.
```

We define here a function $\mathrm{C}(n)$ that converts an element of `N` into a list of bits. It replaces `xH` and `xI` by true and `x0` by false. We also implement the reverse function C', so that $\mathrm{C}'(\mathrm{C}(n)) = n$. Note: the auxiliary function associated to C converts any list with one element into x_{H}, so that C is not injective.

```
Fixpoint list_of_pos p :=
   match p with
   | xO n => false:: list_of_pos n
   | xI n => true:: list_of_pos n
   | xH => [:: true]
end.

Fixpoint pos_of_list l :=
  match l with
    | [ :: _ ] => xH
    | a :: l => if a then xI (pos_of_list  l) else xO (pos_of_list l)
    | nil => xH
end.

Definition list_of_num p :=  if p is Npos p then list_of_pos p else nil.
Definition num_of_list l :=
  if l is a :: b then Npos (pos_of_list l) else 0%num.
```

We show here that $C'(C(n)) = n$ and $C(C'(L)) = L$ whenever L is a list whose tail is true. Conversely, if n is non-zero, the reverse of $C(c(n))$ starts with true. Note that, in any case, $C(C'(L))$ has the same size as L.

```
Lemma list_of_posK: cancel list_of_pos pos_of_list.
Lemma list_of_numK: cancel list_of_num num_of_list.
Lemma pos_of_listK x: list_of_pos (pos_of_list (rcons x true)) = rcons x true.
Lemma num_of_listK x: list_of_num (num_of_list (rcons x true)) = rcons x true.
Lemma num_of_list_false x:
   num_of_list (rcons x false) = num_of_list (rcons x true).
Lemma num_of_list_sizeK x: size (list_of_num (num_of_list x)) = size x.
Lemma nat_list_succ' n (q:= (list_of_num (bin_of_nat n.+1))):
  q = rcons (belast(head true q) (behead q)) true.
Lemma nat_list_succ n (q:= (list_of_num (bin_of_nat n.+1))):
  rev q =  true :: behead  (rev q).
```

Let $C''(L)$ be the function that computes $\sum_i a_i 2^i$ where a_i is the i-th element of L converted to zero or one. Since the leading coefficient may be zero, this function is defined on `nat` and not `N`. Obviously, if the last element of L is false, it can be ignored; if it is true, then $C''(L) = c'(C'(L))$.

```
Fixpoint nat_of_list l :=
  if l is a:: b then let r := (nat_of_list b).*2 in if a then r.+1 else r
  else 0.

Lemma nat_of_list_rF l: nat_of_list l = nat_of_list (rcons l false).
Lemma nat_of_list_rT l:
  list_to_nat (rcons l true) =  nat_of_bin (num_of_list (rcons l true)).
```

We consider here the base two reverse of an integer n. This is $r_2(n) = C''(r(C(c(n))))$ where $r(L)$ is the reverse of L. In the case where n is even, C yields a list that starts with zero, so that C'' is provided a list that ends with zero, which is ignored. It follows that $r_2(2n) = r_2(n)$. Otherwise, it is not ignored so that $r_2(n) = c'(C'(r(C(c(n)))))$. If n is non-zero, C yields a list that ends with true, so that C'' gets a list that starts with true, and returns an odd number. Thus, if n is non-zero, in $r_2(r_2(n))$ we can replace both C'' by C'. All terms simplify, and the result is n; in other terms $r_2(r_2(n)) = n$.

```
Definition base2rev (n:nat) :=
  list_to_nat (rev (list_of_num (bin_of_nat n))).

Lemma base2rev0: base2rev 0 = 0.
Lemma base2rev_even n: base2rev (n.*2) = base2rev n.
Lemma base2rev_odd n: odd n ->
  base2rev n = nat_of_bin (num_of_list (rev (list_of_num (bin_of_nat n)))).

Lemma odd_base2rev n: odd (base2rev n.+1).
Lemma base2rev_oddI n: odd n -> base2rev (base2rev n) = n.
Lemma base2rev_oddII p (n := (base2rev p)): base2rev (base2rev n) = n.
```

1.1.7 Base two logarithm

We define $\log_2(n)$ as the number of bits in the binary expansion of n. This is zero if $n = 0$, non-zero otherwise.

```
Definition log2 n := size (list_of_num (bin_of_nat n)).

Lemma log2_0: log2 0 = 0.
Lemma log2_1: log2 1 = 1.
Lemma log2_eq0 n: (log2 n == 0) = (n == 0).
Lemma log2S_k n: (log2 n.+1) = (log2 n.+1).-1.+1.
```

The number $\log_2(n)$ is the only integer k such that $2^{k-1} \leq n < 2^k$. Thus, $\log_2(2n) = \log_2(n+1) = 1 + \log_2(n)$.

```
Lemma log2_bnd n (k := log2 n.+1): 2^(k.-1) <= n.+1 < 2^k.
Lemma log2_pr n m: 2 ^ m <= n < 2^m.+1 -> log2 n = m.+1.
Lemma log2_double n: log2 ((n.+1).*2) = (log2 n.+1).+1.
Lemma log2_Sdouble n: log2 (n.*2.+1) = (log2 n).+1.
Lemma exp2_log_even n: ~~ odd (2 ^(log2 n.+1)).
Lemma leqn_log m n: m <= n -> log2 m <= log2 n.
Lemma log2_pow n: log2 (2^ n) =  n.+1.
Lemma log2_pred_pow n: log2 ((2^ n).-1) =  n.

Lemma elogn2_unique a b a' b':
   (a.*2.+1) * 2 ^ b =  (a'.*2.+1) * 2 ^ b' -> a = a'.
Lemma log2_mul_exp a n: a != 0 -> log2 (a* 2^n) = n + log2 a.
```

If $k = \log_2(n)$ then $n = 2^{k-1} + r$, where r is the remainder in the division of n by 2^{k-1}. We have $r_2(2n+1) = 2^k + r_2(n)$. Thus, if n is odd, the numbers n and $r_2(n)$ have the same base two logarithm.

```
Lemma log2_pr1 n (q := 2^ (log2 n).-1) :  n !=0 -> n = q + (n %% q).
Lemma base2r_odd n: base2rev (n.*2.+1) = 2^ (log2 n) + base2rev n.
Lemma log_base2r  n: odd n -> log2 (base2rev n) = log2 n.
```

1.1.8 Rings and fields

If we take two integers, convert it to **Q** and add (or multiply), the result is the same as if perform first the result on **N** and convert it. These lemmas are trivial (via a right to left rewrite), but given only the LHS, Coq is unable to find some implicit arguments.

```
Lemma ratN_M (n m: nat) : (n*m)%N%:Q = (n)%:Q * (m)%:Q.
Lemma ratN_D (n m: nat) : (n+m)%N%:Q = (n)%:Q + (m)%:Q.
Lemma intr_N (t: int): (-t)%:Q = -(t%:Q).
```

In these lemmas, all arguments belong to a field.

```
Lemma addf_div1 x y: y != 0 -> 1 + x / y = (x+y) /y.
Lemma subf_div1 x y: y != 0 -> 1 - x / y = (y - x) /y.
Lemma subf_div2 x y: y != 0 -> x / y - 1 = (x - y) /y.
Lemma invf_div x y:  (x / y)^-1 = y /x.
Lemma addf_inv x y: y != 0 -> x + y^-1 = (x*y +1)/y.
Lemma doubleq (x: rat): x *2%:Q = x + x.
```

1.2 Division on Z

Division on **Z** in SSREFLECT depends on lots of files (poly, polydiv, finalg, perm, matrix, mxalgebra, vector, etc). We provide here a standalone version, by copying some definitions and lemmas.

These are the definitions.

```
Definition divz (m d : int) :=
  let: (K, n) := match m with Posz n => (Posz, n) | Negz n => (Negz, n) end in
  sgz d * K (n %/ `|d|)%N.

Definition modz (m d : int) : int := m - divz m d * d.

Infix "%/" := divz : int_scope.
Infix "%%" := modz : int_scope.
```

Here are some properties.

```
Lemma divz_nat (n d : nat) : (n %/ d)%Z = n %/ d.
Lemma divzN m d : (m %/ - d)%Z = - (m %/ d)%Z.
Lemma divz_abs m d : (m %/ `|d|)%Z = (-1) ^+ (d < 0)%R * (m %/ d)%Z.
Lemma div0z d : (0 %/ d)%Z = 0.
Lemma divNz_nat m d : (d > 0)%N -> (Negz m %/ d)%Z = - (m %/ d).+1%:Z.
Lemma divz_eq m d : m = (m %/ d)%Z * d + (m %% d)%Z.
Lemma modzN m d : (m %% - d)%Z = (m %% d)%Z.
Lemma modz_abs m d : (m %% `|d|%N)%Z = (m %% d)%Z.
Lemma modz_nat (m d : nat) : (m %% d)%Z = m %% d.
Lemma modNz_nat m d : (d > 0)%N -> (Negz m %% d)%Z = d%:Z - 1 - (m %% d)%:Z.
Lemma modz_ge0 m d : d != 0 -> 0 <= (m %% d)%Z.
Lemma divz0 m : (m %/ 0)%Z = 0.
Lemma mod0z d : (0 %% d)%Z = 0.
Lemma modz0 m : (m %% 0)%Z = m.
Lemma divz_small m d : 0 <= m < `|d|%:Z -> (m %/ d)%Z = 0.
Lemma divzMDl q m d : d != 0 -> ((q * d + m) %/ d)%Z = q + (m %/ d)%Z.
Lemma mulzK m d : d != 0 -> (m * d %/ d)%Z = m.
Lemma mulKz m d : d != 0 -> (d * m %/ d)%Z = m.
Lemma expzB p m n : p != 0 -> (m >= n)%N ->
  p ^+ (m - n) = (p ^+ m %/ p ^+ n)%Z.
Lemma modz1 m : (m %% 1)%Z = 0.
Lemma divn1 m : (m %/ 1)%Z = m.
```

```
Lemma divzz d : (d %/ d)%Z = (d != 0).
Lemma ltz_pmod m d : d > 0 -> (m %% d)%Z < d.
Lemma ltz_mod m d : d != 0 -> (m %% d)%Z < `|d|.
Lemma divzMpl p m d : p > 0 -> (p * m %/ (p * d) = m %/ d)%Z.
Lemma divzMpr p m d : p > 0 -> (m * p %/ (d * p) = m %/ d)%Z.
Lemma lez_floor m d : d != 0 -> (m %/ d)%Z * d <= m.
Lemma lez_div m d : (`|(m %/ d)%Z| <= `|m|)%N.
Lemma ltz_ceil m d : d > 0 -> m < ((m %/ d)%Z + 1) * d.
Lemma ltz_divLR m n d : d > 0 -> ((m %/ d)%Z < n) = (m < n * d).
Lemma lez_divRL m n d : d > 0 -> (m <= (n %/ d)%Z) = (m * d <= n).
Lemma divz_ge0 m d : d > 0 -> ((m %/ d)%Z >= 0) = (m >= 0).
Lemma divzMA_ge0 m n p : n >= 0 -> (m %/ (n * p) = (m %/ n)%Z %/ p)%Z.
Lemma modz_small m d : 0 <= m < d -> (m %% d)%Z = m.
```

1.3 The floor function on Q

We define here the floor function $\lfloor x \rfloor$ on **Q**, and prove some basic properties. Note that $\lfloor a/b \rfloor$ is the quotient of a by b when $b > 0$ (since reducing the fraction amounts to divide by a positive number).

```
Definition floorq x := ((numq x) %/ (denq x))%Z.

Lemma succq (z: int): z%:Q +1 = (z+1) %:Q.
Lemma floorp1 x (y:= (floorq x)%:Q): y <= x < y+1.
Lemma floorp2 x (y: int): y%:Q <= x < y%:Q +1 ->  y = floorq x.
Lemma floor_ge0 x: 0 <= x -> 0 <= floorq x.
Lemma floor_small x: 0 <= x < 1 -> floorq x = 0.
Lemma floor_div (a b:int): 0 < b -> floorq (a%:Q / b%:Q) = (a %/b)%Z.
Lemma floor_sum (a:rat) (b: int):  floorq (a + b%:Q) = floorq a + b.
Lemma floor_succ a : floorq (a + 1) = (floorq a) + 1.
```

We consider here the functions

$$f(x) = \frac{1}{1+2\lfloor x \rfloor - x}, \qquad g(x) = \frac{1}{-x} - 1 - 2\lfloor \frac{1}{-x} \rfloor. \tag{1.5}$$

These are inverse one of the other. We shall give a short name `Sn` to the first function.

```
Definition Stern_succ_fct x := (1 + (floorq x)%:Q *2%:Q - x)^-1.
Definition Stern_prev_fct x  (y := (- x)^-1) :=
  y - 1 - (floorq y)%:Q * 2%:Q.

Notation Sn := Stern_succ_fct.

Lemma Stern_succ_fctK: cancel Sn  Stern_prev_fct.
Lemma Stern_prev_fctK: cancel Stern_prev_fct Sn.
```

The denominator of f is $\lfloor x \rfloor + (1 + \lfloor x \rfloor - x)$, so that it is > 0 when $\lfloor x \rfloor \geq 0$, i.e., $x \geq 0$. If $0 \leq x < 1$, the function simplifies to $f(x) = (1-x)^{-1}$. If $\mathrm{T}(x) = -1/x$, then $f(\mathrm{T}(f(x))) = \mathrm{T}(x)$, whenever $x > 0$. We have the following relations:

$$f(\frac{a}{a+b}) = \frac{a+b}{b}, \qquad f(\frac{a+b}{b}) = \frac{b}{a+b-2(a \bmod b)}. \tag{1.6}$$

```
Lemma Sn_0: Sn 0 = 1.
Lemma Sn_1: Sn 1 = 1/(2%:Q).

Lemma Sn_gt0 x: 0 <= x -> 0 < (Sn x).
Lemma Sn_small x: 0 <= x < 1 -> (Sn x) = (1-x) ^-1.
Lemma Sn_neg x  (T:= fun x => (- x)^-1):
  0 <x  -> Sn (T (Sn x)) = T x.

Lemma Sn_lt1 (a b: int)  (A := a%:Q) (B := b%:Q):
  0 <= a -> 0 < b  -> Sn (A / (A + B)) = (A + B) / B.
Lemma Sn_gt1 (a b: int) (c := (a + b) - (a %% b)%Z *2)
  (A := a%:Q) (B := b%:Q)  (C := c%:Q):
  0 <= a -> 0 < b -> Sn ((A + B) / B) = B / (B + C).
```

1.4 Continued fractions

A continued fraction is the expression $a_0 + b_1/(a_1 + b_2/(a_2 + \ldots))$; for instance if $a_0 = 3$, $a_i = 6$ for non-zero i and $b_i = (2i-1)^2$ one gets π. In what follows, we consider only the case where $b_i = 1$, these are *simple continued fraction*, and satisfy $C(x :: l) = x + 1/C(l)$. In the case where L is infinite, one may give a sense to C(L) by truncating L to its first n terms, say L_n, and taking the limit of $C(L_n)$. It exists under some conditions (for instance when each element of the list is real and ≥ 1, in particular when each element is a non-zero integer, but the theory can be applied when the list contains complex numbers). Proving that the limit satisfies the recurrence given above is non-trivial. On the other hand, if the list is finite, the question is how to finish the recurrence. For instance $C([a,b]) = a + 1/C(b) = a + 1/b$. So, in $C(x :: l) = x + 1/C(l)$, if l is empty, one wants $1/C(l)$ to be zero.

We avoid division by zero by defining $C'(L) = C(0 :: L)$ by induction via $C'(x :: l) = 1/(x + C'(l))$ and $C'(\emptyset) = 0$. Now, if every element of L is > 0 then $C'(L) > 0$ (unless L is empty). We then define $C(x :: L)$ to be $x + C(L)$. If all elements of L are > 0 this is ≥ x. The definition is completed by saying that $C'(\emptyset) = 0$.

The formula for $C(L + +L')$ is obvious by induction. It implies that

$$C(a_0, a_1, \ldots, a_{n-1}, a_n, a_{n+1}) = C(a_0, a_1, \ldots, a_{n-1}, a_n + 1/a_{n+1})$$

```
Implicit Type L : seq rat.

Fixpoint SCF' L:= if L is (a::l) then 1 / (a + SCF' l) else 0.
Definition SCF L := if L is a::l then a + SCF' l else 0.

Lemma SCF_pos1 L : all (fun z => 0 < z) L -> L != nil -> 0 <  SCF' L.
Lemma SCF_pos2 a L : all (fun z => 0 < z) L ->  a <= SCF (a::L).

Lemma SCF1 a  : SCF [:: a ] = a.
Lemma SCF2 a b : SCF [:: a ; b ] =  a + 1 / b.
Lemma SCF_rec a l : SCF (a :: l) = a + 1 / SCF l.
Lemma SCF_cat L1 L2 : L1 != nil -> L2 != nil ->
   SCF (L1 ++ L2) = SCF(rcons L1 (SCF L2)).
Lemma SCF_recl L a b: SCF (rcons (rcons L a) b) =  SCF (rcons L (a+1/b)).
```

1.5 The Fibonacci sequence

The Fibonacci sequence is defined in a file (written by L. Thery) that is not part of the standard SSREFLECT library (but by in the CoqFinitGroup repository). We provide a copy of some definitions and lemmas. The only missing property is a matrix identity of the form

$$Q^{p/r} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{p/r} = \begin{pmatrix} F_{(p+r)/r} & F_{p/r} \\ F_{p/r} & F_{(p-r)/r} \end{pmatrix}.$$

The library proves the relation for $r = 1$ and $p > 0$. Here $F_k = (\alpha^k - \beta^k)/(\alpha - \beta)$, where $\alpha(1 + \sqrt{5})/2$, $\beta(1 - \sqrt{5})/2$. Since $\beta < 0$, the quantity $F_{p/r}$ is in general not real. However, for $r = 1$, it is an integer and satisfies a recurrence relation, which can be used to define it.

```
Fixpoint  fib_rec n :=
  if n is n1.+1 then
    if n1 is n2.+1 then fib_rec n1 + fib_rec n2
    else 1
  else 0.

Definition fib := nosimpl fib_rec.

Lemma fibE : fib = fib_rec.
Lemma fib0 : fib 0 = 0.
Lemma fib1 : fib 1 = 1.
Lemma fibSS n: fib n.+2 = fib n.+1 + fib n.
Lemma fib_gt0 m: 0 < m -> 0 < fib m.
Lemma fib_smonotone m n: 1 < m < n -> fib m < fib n.
Lemma fib_monotone m n: m <= n -> fib m <= fib n.
Lemma fib_eq1 n: (fib n == 1) = ((n == 1) || (n == 2)).
Lemma fib_eq m n:
  (fib m == fib n) = [|| m == n, (m == 1) && (n == 2) | (m == 2) && (n == 1)].
```

There is a formula for F_{a+b}, and one for F_{a-b} that depends on the parity of b.

```
Lemma fib_add m n:
  m != 0 ->  fib (m + n) = fib m.-1 * fib n + fib m * fib n.+1.
Lemma fib_sub m n: n <= m ->
   fib (m - n) = if odd n then fib m.+1 * fib n - fib m * fib n.+1
                 else fib m * fib n.+1 - fib m.+1 * fib n.
Lemma fib_doubleS n: fib (n.*2.+1) = fib n.+1 ^ 2 + fib n ^ 2.
Theorem dvdn_fib m n: m %| n -> fib m %| fib n.
Lemma fib_prime p: p != 4 -> prime (fib p) -> prime p.
```

We give some formulas involving sums of Fibonacci numbers. On the other hand, the sum of the binomial coefficients $\binom{n}{k}$, where the sum $n+k$ is constant, is a Fibonacci number; we shall consider a variant of this below.

```
Lemma fib_sum n: \sum_(i < n) fib i = (fib n.+1).-1.
Lemma fib_sum_even n: \sum_(i < n) fib i.*2 = (fib n.*2.-1).-1.
Lemma fib_sum_odd n: \sum_(i < n) fib i.*2.+1 = fib n.*2.
Lemma fib_sum_square n: \sum_(i < n) (fib i)^2 = fib n * fib n.-1.
Lemma bin_sum_diag n: \sum_(i < n) 'C(n.-1-i,i) = fib n.
```

Lucas numbers. They are defined by $L_0 = 2$, $L_1 = 1$, and satisfy the same recurrence relation as the Fibonacci numbers. Note that $L_n = F_{n-1} + F_{n+1}$.

```
Fixpoint lucas_rec n :=
  if n is n1.+1 then
    if n1 is n2.+1 then lucas_rec n1 + lucas_rec n2
    else 1
  else 2.
Definition lucas := nosimpl lucas_rec.

Lemma lucasE : lucas = lucas_rec.
Lemma lucas0 : lucas 0 = 2.
Lemma lucas1 : lucas 1 = 1.
Lemma lucasSS n: lucas n.+2 = lucas n.+1 + lucas n.
Lemma lucas_fib n: n != 0 -> lucas n = fib n.+1 + fib n.-1.
Lemma lucas_gt0 m: 0 < lucas m.
Lemma double_lucas n: 3 <= n -> (lucas n).*2 = fib (n.+3) + fib (n-3).
Lemma fib_double_lucas n: fib (n.*2) = fib n * lucas n.
```

Some easy properties. What follows is not part of the *fib.v* file. Let's say that a sequence is "like fib" when $x_{n+2} = x_{n+1} + x_n$ (for instance, Fibonacci and Lucas sequences are like fib). This operation is linear and shift invariant. Such sequence is uniquely determined by the first two terms.

```
Definition like_fib F := (forall n, F n.+2 = F n.+1 + F n).

Lemma fib_like_fib: like_fib fib.
Lemma lucas_like_fib: like_fib lucas.
Lemma like_fib_mul F c: like_fib F -> like_fib (fun n => c * F n).
Lemma like_fib_add F F': like_fib F -> like_fib F' ->
  like_fib (fun n => F n + F' n).
Lemma like_fib_shift F m: like_fib F -> like_fib(fun n => F (n + m)).
Lemma like_fib_succ F: like_fib F -> like_fib(fun n => F (n.+1)).
Lemma like_fib_eq F F':
  like_fib F -> like_fib F' -> F 0 = F' 0 -> F 1 = F' 1 -> F =1 F'.
Lemma is_fib F: like_fib F -> F 0 = 0 -> F 1 = 1 -> F =1 fib.
```

If (x_n) is like fib, one has $x_n = x_1 \mathrm{F}_n + x_0 \mathrm{F}_{n-1}$. This implies

$$x_{n+m+1} = x_{n+1}\mathrm{F}_{m+1} + x_n \mathrm{F}_m.$$

One deduces a formula for F_{n+m} (given above) and for L_{n+m}. One has

$$5x_n = (3x_0 - x_1)\mathrm{L}_n + (2x_1 - x_0)\mathrm{L}_{n+1}$$

(note that x_0/x_1 has to be between $1/3$ and 2 in order for the two coefficients to be ≥ 0). Example

$$5\mathrm{F}_{n+2} = 3\mathrm{L}_{n+1} + \mathrm{L}_n \text{ and } 5\mathrm{F}_{n+1} = \mathrm{L}_{n+1} + 2\mathrm{L}_n.$$

Note that we have also $2x_n = x_0\mathrm{L}_n + (2x_1 - x_0)\mathrm{F}_n$ (if y and y' are linearly independent fib like sequences, we can always write $x_n = \alpha y_n + \beta y'_n$ where α and β are obtained by considering the special cases $n = 0$ and $n = 1$). The formula for F_{2n} is the same as that given above, with L_n replaced by its value; the formula for L_{2n} will be generalized below:

$$\mathrm{F}_{2n} = \mathrm{F}_n(\mathrm{F}_n + 2\mathrm{F}_{n-1}), \qquad \mathrm{L}_n^2 = \mathrm{L}_{2n} + 2\cdot(-1)^n.$$

```
Lemma like_fibE F: like_fib F ->
   forall n, F n.+1 = (F 0) * (fib n) + (F 1) * (fib n.+1).
Lemma like_fib_shiftE F m n: like_fib F ->
  F (n+m).+1 = F (n.+1)* fib m.+1 + F n * fib m.

Lemma lucasS n: lucas n.+1 = fib n.+1 + (fib n).*2.
Lemma lucas_add m n: lucas (n+m).+1 = lucas (n.+1)* fib m.+1 + lucas n * fib m.

Lemma like_fib_lucas F: like_fib F -> F 0 <= 2 * F 1 -> F 1 <=  3 * (F 0) ->
   forall n, 5 * F n =
     (3* (F 0) - F 1) * (lucas n) + (2* (F 1) - F 0)* (lucas n.+1).
Lemma like_fib_bis F: like_fib F -> F 0 <= 2 * F 1 ->
   forall n, 2 * F n = (F 0) * (lucas n) + (2* (F 1) - F 0)* (fib n).
Lemma lucas_fib2 n: 5 * fib n.+2 = 3 * lucas n.+1 + lucas n.
Lemma lucas_fib3 n: 5 * fib n.+1 = lucas n.+1 + (lucas n).*2.
Lemma lucas5S n: lucas (n+5) = 5 * lucas n.+1 + 3 * lucas n.
Lemma fib3S n: fib n.+3 = (fib n.+1).*2 + (fib n).
Lemma fib4S n: fib n.+4 = 3*(fib n.+1) + (fib n).*2.
Lemma fib_square_succ n: (fib n.+2)^2 = (fib n.+1)^2 + (fib n)*(fib n.+3).
Lemma fib_double n: fib ((n.+1).*2) = (fib n.+1) *((fib n).*2 + fib (n.+1)).
Lemma fib_square_n3_n n: fib n.+3 ^ 2 + fib n ^ 2 = (fib (n.+1).*2.+1).*2.
Lemma lucas_square n:
  (lucas n)^2 = (if (odd n) then subn else addn) (lucas (n.*2)) 2.
```

Note that F_n is zero only when $n = 0$ and is $\geq n-1$ (the relation $n \leq F_n$ holds but for $n = 2$), and if $n \geq 3$ then $F_n \geq 2$, so that $F_n = F_{n-2}+2$. We shall use later one the (trivial) property that if $a \leq F_{n+1}$, $b \leq F_n$ and $a + b = F_{n+2}$ then $a = F_{n+1}$ and $b = F_n$. For every x there is a unique a such that $F_a \leq x < F_{a+1}$ (we show here uniqueness, existence will be considered later on).

```
Lemma fib_pos n: fib (n.+1) = (fib (n.+1)).-1.+1.
Lemma fib_eq0 n: (fib n == 0) = (n == 0).
Lemma fib_gen n: n <= fib n.+1.
Lemma fib_monotone_bis a b: fib a < fib b -> a < b.
Lemma fib_smonotone_bis a b: a < b -> fib a.+2 < fib b.+2.
Lemma fib_ge2_alt n: fib n.+3 = (fib n.+3).-2.+2.
Lemma fib_sum_bound a b n: a <=  fib n.+1 -> b <=  fib n ->
   a + b = fib n.+2  -> (a = fib n.+1) /\ (b = fib n).
Lemma fib_partition a b x:
   fib a <= x < fib a.+1 -> fib b <= x < fib b.+1  -> a = b.
```

The function L_n is strictly increasing for $n > 0$; it is injective.

```
Lemma lucas_smonotone m n: 0 < m < n -> lucas m < lucas n.
Lemma lucas_monotone m n: 0 < m <= n -> lucas m <= lucas n.
Lemma lucas_monotone_bis m n: lucas m <= lucas n -> ((m <= n) || (n == 0)).
Lemma lucas_injective: injective lucas.
Lemma lucas_injective_bis a b: (lucas a == lucas b) = (a== b).
Lemma lucas_smonotoneE m n:
  lucas m < lucas n = [|| (0< m < n), (m==1) && (n== 0)| (m == 0) && (2<= n )].
Lemma lucas_monotoneE m n:
  lucas m <= lucas n =
    [|| m== n, 0< m < n, (m==1) && (n== 0)| (m == 0) && (2<= n )].
```

We show here that $\gcd(F_n, F_m) = F_{\gcd(n,m)}$ (note: the SSREFLECT library has a proof that use the value of F_{n-m}). We may always assume $0 < n < m$ (by symmetry, the case $n = m$

being trivial). Write $n = d + m$, so that $F_n = F_{d-1}F_m + F_dF_{m+1}$. When we compute the gcd with F_m we can ignore the first term, and the factor of F_d (note that F_m is coprime to F_{m+1}, by induction). Now in the right has side, we replace $\gcd(n,m)$ by $\gcd(r,m)$ and conclude by induction on $\max(n,m)$.

```
Lemma coprimeSn_fib n: coprime (fib n) (fib n.+1).
Lemma gcd_fib n m:gcdn (fib n) (fib m) = fib (gcdn n m).
```

We show here that F_n is even if and only if n is a multiple of three. The same holds for L_n as $L_n = F_n + 2F_{n-1}$. Note: if $x_{n+2} = ax_{n+1} + bx_n$ where a and b are odd, then either every x_i is even or one ouf of three is even. From $F_{n+3} = 2F_{n+1} + F_n$ one deduces that $F_{n+3} = F_n$ modulo 2, so that, to check whether F_n is even, it suffices to consider $n < 3$. A shorter proof is: let $g = \gcd(F_n, 2) = \gcd(F_n, F_3) = F_{\gcd(n,3)} = F_{\gcd(r,3)}$, where r is the remainder in the division of n by 3. In case $r = 1$ or $r = 2$, $q = \gcd(r,3)$ is trivial so that $g = 1$; otherwise $q = 3$ and $g = 2$; finally F_n odd means $g = 1$.

From $F_{n+4} = 3F_{n+1} + 2F_n$ one deduces that 3 divides F_n if and only if n is a multiple of 4. From $L_{n+2} = 3F_{n+1} + F_n$, one deduces that 3 divides L_n if and only if n is 2 modulo 4.

From $L_n = F_n + 2F_{n-1}$ one deduces that $\gcd(L_n, F_n) = F_{\gcd(n,3)}$. This quantity is 2 if n is a multiple of 3, and 1 otherwise.

```
Lemma fib_is_even_mod3 n: odd (fib n)  =  (n %%3 !=0).
Lemma lucas_is_even_mod3 n: odd (lucas n)  =  (n %%3 !=0).

Lemma fib_mod3 n: 3 %| (fib n) =  (4 %| n).
Lemma lucas_mod3 n: 3 %| (lucas n) = (n== 2 %[mod 4]).

Lemma gcd_lucas_fib1 n: gcdn (lucas n.+1) (fib n.+1) = fib (gcdn n.+1 3).
Lemma gcd_lucas_fib2 n:  (n %%3 ==0) -> gcdn (lucas n) (fib n) = 2.
Lemma gcd_lucas_fib3 n: (n %%3 !=0) -> coprime (lucas n) (fib n).
```

We have

$$\sum_{i=0}^{n}\binom{n}{i}F_i = F_{2n}, \qquad \sum_{i=0}^{n}\binom{n}{i}F_{i+1} = F_{2n+1}.$$

```
Lemma sum_fib_pascal n:
  \sum_(i<n.+1) 'C(n,i)* fib i = fib(n.*2) /\
  \sum_(i<n.+1) 'C(n,i)* fib i.+1 = fib(n.*2.+1).
```

We give here some inequalities involving Lucas and Fibonacci numbers.

```
Lemma fib_lt_lucas n: n != 0 -> fib n <= lucas n ?= iff (n==1).
Lemma lucas_lt_fib n: n != 0 -> lucas n <= fib n.+2 ?= iff (n==2).
Lemma lucas_ge_2fib n: 2 <= n -> (fib n).*2 <= lucas n.
Lemma lucas_monotone2 n p: p <= n -> lucas (n - p) <= lucas (n + p).
Lemma fib_mon1 n: 2 <= n -> 3* fib n <= 2 * fib n.+1.
Lemma lucas_mon1 n: 3 <= n -> 3* lucas n <= 2 * lucas n.+1.
Lemma fib_square_bound n: 2 <= n -> fib n.*2 < (fib n.+1)^2 < fib (n.*2.+1).
Lemma lucas_square_bound n: 3 <= n ->
  fib n.*2.+1 < (lucas n)^2 < fib (n.*2.+2).
Lemma fib_lt_exp n: 3 <= n -> fib n.+2 < 2^n.
```

The two quantities F_{a+b} and F_{a-b} are of the form $u+v$ and $w \pm v$. If one eliminates v, the result is a Fibonacci number times a Lucas number:

$$L_j F_k = F_{k+j} \pm F_{k-j} \text{ (plus when } j \text{ even)} \quad (j \le k) \tag{1.7}$$

$$L_j F_k = F_{k+j} \pm F_{j-k} \text{ (plus when } k \text{ odd)} \quad (k \le j). \tag{1.8}$$

It follows

$$L_n L_p = L_{n+p} \pm L_{n-p} \text{ (plus when } p \text{ even)} \tag{1.9}$$

$$5F_n F_p = L_{n+p} \pm L_{n-p} \text{ (plus when } p \text{ odd)}. \tag{1.10}$$

Adding the two relations gives

$$2F_{n+p} = L_n F_p + L_p F_b, \tag{1.11}$$

$$2L_{n+p} = L_n L_p + 5F_n F_p. \tag{1.12}$$

```
Lemma fib_lucas1 j k: j <= k ->
  lucas j * fib k =
    (if (odd j) then subn else addn) (fib (k + j)) (fib (k - j)).
Lemma fib_lucas2 j k: k <= j ->
  lucas j * fib k =
    (if (odd k) then addn else subn) (fib (k + j)) (fib (j - k)).
Lemma lucas_lucas1 p n: p <= n ->
  lucas n * lucas p =
    (if (odd p) then subn else addn) (lucas (n + p)) (lucas (n - p)).
Lemma lucas_lucas2 p n: p <= n ->
  5 * fib n * fib p =
    (if (odd p) then addn else subn) (lucas (n + p)) (lucas (n - p)).
Lemma lucas_lucas4 n p:
  (fib (n + p)).*2 = lucas n * fib p +  lucas p  * fib n.
Lemma lucas_lucas3 n p:
  (lucas (n + p)).*2 = lucas n * lucas p + 5 * fib n * fib p.
Lemma lucas_n_fib_mod5 n: n * lucas n == fib n %[mod 5].
```

1.5.1 A Diophantine Equation

Consider

$$c^2 = d^2 + cd + 1, \tag{1.13}$$

$$c^2 + 1 = d^2 + cd. \tag{1.14}$$

```
Definition DE_eq1 c d := c^2 = d^2 + c*d + 1.
Definition DE_eq2 c d := c^2 +1 = d^2 + c*d.
```

If (c,d) is a solution of the second equation, so is $(2c+d, c+d)$. Moreover, obviously, $d > 0$ and if $0 < c$ then $d \le c < 2d$, so that $(c-d, 2d-c)$ is another solution. Finally, if $c = 0$, then d has to be 1; it follows by induction that (F_{2n}, F_{2n-1}) is a solution and that all solutions are of this form, with the convention that $F_{-n} = (-1)^{n+1} F_n$, so that $F_{-1} = 1$.

```
Lemma DE2_rec c d:  DE_eq2 c d <-> DE_eq2 (c.*2 +d) (c+d).
Lemma DE2_lt0d c d: DE_eq2 c d -> 0 <d.
Lemma DE2_pos c d: 0 < c -> DE_eq2 c d -> d <= c < d.*2.
Lemma DE2_rec' c d: 0 < c -> DE_eq2 c d -> DE_eq2 (c-d) (d.*2-c).
Lemma DE2_solution c d: DE_eq2 c d <->
  (c=0 /\ d = 1) \/ exists n, c = fib n.*2.+2 /\ d = fib n.*2.+1.
```

If (c,d) is a solution of (1.13), then $(c+d,c)$ is a solution of (1.14) and conversely. So the solutions of (1.13) are $c = F_{2n+1}$ and $d = F_{2n}$.

```
Lemma DE_equiv_12 c d: DE_eq1 c d <-> DE_eq2 (c+d) c.
Lemma DE1_solution c d: DE_eq1 c d <->
   exists n, c = fib n.*2.+1 /\ d = fib n.*2.
```

In particular

$$F_{2n+1}^2 = F_{2n}^2 + F_{2n+1}F_{2n} + 1, \qquad F_{2n+2}^2 = F_{2n+1}^2 + F_{2n+2}F_{2n+1} - 1.$$

We also have $F_{m+3}^2 + F_m^2 = 2F_{2m+3}$.

```
Lemma DE1_sol n:
   (fib n.*2.+1)^2 = (fib n.*2)^2 + (fib n.*2.+1)*(fib n.*2) + 1.
Lemma DE2_sol n:
   (fib n.*2.+2)^2 + 1 = (fib n.*2.+1)^2 + (fib n.*2.+2)*(fib n.*2.+1).
Lemma fib_square_n3_n n:
   fib n.+3 ^ 2 + fib n ^ 2 = (fib (n.+1).*2.+1).*2.
```

We consider here

$$(y \pm 1)^2 + (2y)^2 = z^2. \tag{1.15}$$

Consider first the equation with $y+1$. We pretend that the solutions are $y = F_{2n}F_{2n+1}$ and $z = F_{4n+1}$. It is easy to check, thanks to (1.13), that this is indeed a solution. On the other hand, consider a solution (y,z). There are three numbers p, q and r such that p and q are coprime, $q \le p$ and either $y+1 = r(p^2-q^2)$ and $2y = 2rpq$ or $2y = r(p^2-q^2)$ and $y+1 = 2rpq$. In the first case r divides y and $y+1$ so $r = 1$. Now, (p,q) satisfies (1.13), so that there is an integer n such that $p = F_{2n+1}$ and $q = F_{2n}$, and we can conclude. In the second case, y is odd as $y+1 = 2rpq$. Thus $\gcd(2y, y+1) = 2$. Moreover p^2-q^2 and $2pq$ are coprime, so that $r = 2$. Set $p' = p+q$ and $q' = p-q$. The relation $2y = r(p^2-q^2)$ says $y = p'q'$, and $y+1 = 2rpq$ says $y+1 = p'^2 - q'^2$, so that (p',q') satisfies (1.13). The conclusion is the same.

Consider now the equation with $y-1$. The reasoning is the same, with (1.14) instead of (1.13). Solutions are $y = F_{2n+1}F_{2n+2}$, $z = F_{4n+3}$.

```
Definition DE_eq3 y z := (z^2 == (y.+1)^2 + (y.*2)^2).
Definition DE_eq4 y z := (z^2 == (y.-1)^2 + (y.*2)^2).

Lemma DE_eq3_solution y z: DE_eq3 y z ->
   (exists n, y = fib n.*2 * fib n.*2.+1 /\ z = fib (n.*2).*2.+1).
Lemma DE_eq4_solution y z: 0 < y ->
  (DE_eq4 y z <->
   exists n, y = fib n.*2.+2 * fib n.*2.+1 /\ z  = fib (n.*2).*2.+3).
```

Now consider (variables are in **Z**)

$$x^2 + xy + x - y^2 = 0, \tag{1.16}$$

$$x^2 - xy - x - y^2 = 0. \tag{1.17}$$

We pretend that any solution (y,x) of (1.17) has the form $(F_{2n}F_{2n+1}, F_{2n+1}^2)$, $(F_{2n}F_{2n+1}, -F_{2n}^2)$, $(-F_{2n+2}F_{2n+1}, F_{2n+1}^2)$ or $(-F_{2n+2}F_{2n+1}, -F_{2n+2}^2)$, for some integer n. That these pairs are solutions follows directly from (1.13) and (1.14).

```
Lemma DE_eq6_sol n (x := fib n.*2.+1) (y := fib n.*2)(z := fib n.*2.+2):
  [/\  (x ^ 2)^2 = (x ^ 2)*(y * x) + (x ^ 2) + (y * x)^2,
       (y ^ 2)^2 + (y ^ 2)*(y * x) + (y ^ 2) = (y * x)^2,
   (x ^ 2)^2 + (x ^ 2)*(z * x) = (x ^ 2) + (z * x)^2 &
   (z ^ 2)^2 +(z ^ 2) = (z ^ 2)*(z * x) +  (z * x)^2] %N.
```

The two equations are similar, it suffices to replace x by $-x$. The second equation is equivalent to $(y+1)^2+(2y)^2=(2x-y-1)^2$.

```
Definition DE_eq5 (x y:int):= (x^+2 + x*y + x - y^+2 == 0).
Definition DE_eq6 (x y:int):= (x^+2 - x*y - x - y^+2 == 0).

Lemma DE_equiv_56 x y: (DE_eq5 x y) = (DE_eq6 (-x) y).
Lemma DE_eq6_alt x y:
  (DE_eq6 x y) = ((x*+2 - (y+1))^+2 == (y+1)^+2 + (y*+2)^+2).
```

Assume $y \geq 0$, and let z be the absolute value of $2x-y-1$. Now (y,z) is a solution of (1.15) (with a plus sign) and there is an integer n such that $y = F_{2n}F_{2n+1}$ and $z = F_{4n+1}$. Since $2x-y-1 = \pm z$, one deduces from (1.13) that $x = F_{2n+1}^2$ or $x = -F_{2n}^2$. If $y<0$ then $(-y,z)$ is a solution of (1.15) (with a minus sign). If follows that for some n, $y = -F_{2n+2}F_{2n+1}$; using (1.14) shows $x = F_{2n+1}^2$ or $x = -F_{2n+2}^2$.

```
Lemma DE_eq6_pos x y: 0 <= y -> DE_eq6 x y ->
  exists n, y = (fib n.*2 * fib n.*2.+1)%N /\
     (x = (fib n.*2.+1 ^ 2)%N \/ x = -((fib n.*2 ^ 2)%N:int)).
Lemma DE_eq6_neg x y: y <0 -> DE_eq6 x y ->
  exists n, y = -((fib n.*2.+2 * fib n.*2.+1)%N:int) /\
     (x = (fib n.*2.+1 ^ 2)%N \/ x = -((fib n.*2.+2 ^ 2)%N:int)).
Lemma DE_eq6_solution (x y:int): (x^+2 - x*y - x - y^+2 == 0) <-> exists n,
  [\/ y = (fib n.*2 * fib n.*2.+1)%N /\ x = (fib n.*2.+1 ^ 2)%N,
      y = (fib n.*2 * fib n.*2.+1)%N /\ x = -((fib n.*2 ^ 2)%N:int),
    y = -((fib n.*2.+2 * fib n.*2.+1)%N:int) /\ x = (fib n.*2.+1 ^ 2)%N |
    y = -((fib n.*2.+2 * fib n.*2.+1)%N:int) /\ x = -((fib n.*2.+2^2)%N:int)
  ].
```

1.5.2 Some relations between Fibonacci and Lucas numbers

This is inspired from [4].

Consider the relation $F_a = F_b + F_c$. By symmetry, we may assume $b \leq c$. We have the generic solution $c = b+1$ and $a = b+2$, and some exceptions since $F_0 = 0$ and $F_1 = F_2$.

We deduce that, in general, a Fibonacci number is not a Lucas number; exceptions are $L_0 = F_3 = 3$, $L_1 = F_1 = F_2 = 1$ and $L_2 = F_4 = 3$.

Consider the relation $L_a = L_b + L_c$. The non-trivial solutions are $L_3 = L_0 + L_0$ and $L_0 = L_1 + L_1$.

```
Lemma fib_plus_fib a b c: b <= c ->
  (fib a == fib b + fib c) =
  [ || (b == 0) && (fib a == fib c),
      [&& b ==c, (c==1) || (c== 2) & a==3],
      [&& b ==1, c==3 & a==4] |
      (c==b.+1) && (a==b.+2)].
```

```
Lemma lucas_plus_lucas a b c: b <= c ->
  (lucas a == lucas b + lucas c) =
     [ || [&& a==3, b==0 & c==0], [&& a==0, b==1 & c==1] |
        (c==b.+1) && (a==b.+2)].
Lemma fib_eq_lucas m n: (fib m == lucas n) =
  [|| (n==0) &&(m==3), (n==1) &&((m==1) || (m==2))  | (n==2)&&(m==4) ].
```

Consider now the equation $F_n = L_k F_m$. If $k = 1$, this is $F_n = F_m$; if $k = 0$, this is $F_n = F_m + F_m$; we have shown above when these conditions hold. If $m = 0$, then $F_m = 0$, so we have the solution $n = 0$, k arbitrary; if $m = 1$ or $m = 2$ we get $F_n = L_k$, see above. In the case $k \geq 2$, $m \geq 3$, we have $F_{2n} = L_n F_n$. There is no other solution.

Consider now $F_n = L_k L_m$. Without loss of generality, we may assume $k \leq m$. We have $F_3 = L_0 L_1$, $F_6 = L_0 L_3$ (since $L_0 = F_3$ and the previous result), $F_1 = F_2 = L_1 L_1$, $F_4 = L_1 L_2$ (since $L_1 = 1$) and $F_8 = L_2 L_4$. There is no other solution. We may assume $k \geq 2$. We start with the case k even, so that $F_n = L_{m+k} + L_{m-k}$. Since $m + k$ is non-zero, the first Lucas number is $F_{m+k-1} + F_{m+k+1}$; we deduce $n > m + k + 1$ as the second Lucas number is non-zero. It follows $F_{m+k-2} \leq L_{m-k}$. The relation $L_{m-k} \leq F_{m-k+2}$ gives a bound on k. As k is even and non-zero, if follows easily $k = 2$. We consider now the case k odd. We have $F_n + L_{m-k} = L_{m+k}$. One can exclude the cases $m = k$, $m = k + 1$ and $m = k + 2$; it follows $L_{m-k} < F_{m-k} + 2$. On the other hand, we have $n \leq m + k + 1$, so after some simplifications $F_{m+k-1} \leq L_{m-k}$. The result follows.

```
Lemma lucas_times_fib_is_fib m n k:
   (fib n == lucas k * fib m) =
   [|| [&& k==0, n==3 & (m==1) || (m==2) ],
       (k==1) && [|| n == m, (n == 1) && (m == 2) | (n == 2) && (m == 1)] ,
       ( ((m==0) && (n == 0))
         || [&& ((m==1) || (m==2)), (k==2) &(n==4) ]) |
       (k==m) && (n == m.*2)].
Lemma lucas_times_lucas_is_fib m n k: k <= m ->
   (fib n == (lucas m) * (lucas k)) =
    [|| [&& k==0, m == 1 & n == 3], [&& k==0, m == 3 & n == 6],
        (k==1) &&
         [|| (m == 1) && (n == 1), (m == 1) && (n == 2) | (m == 2) && (n == 4)]
        | [&& k ==2, m==4 & n==8] ].
```

Consider now the equation $F_n = cF_k$. If $k = 0$, this gives $F_n = 0$; if $k = 1$ or $k = 2$, this gives $F_n = c$. So assume $k > 2$. By computing gcds, it is obvious that k divides n. If $c = 0$, we get $n = 0$, if $c = 1$, we get $F_n = F_k$; so let's assume $2 \leq c$. If $n = rk$, we have $r \geq 2$, so that $n \geq 2k$ and $L_k \leq c$. This means that given c, there is only a finite number of solutions (for any k, there is a unique n). If $c = 2$ or $c = 3$, there is no solution $k > 2$ (note that c is a Lucas number). There is no solution $c \geq 4$ when c is a Fibonacci number (the formula for F_{a+b} shows $F_{n-m+1}F_m < F_n < F_{n-m+2}F_m$).

```
Lemma fib_times_c_is_fib1 n k c: k !=1 -> k != 2 ->
    fib n = c * (fib k) -> k %| n.
Lemma fib_times_c_is_fib2 n k c: 2 <= c -> 2 < k ->
    fib n = c * (fib k) -> (lucas k) <= c.
Lemma fib_times_c_is_fib3 n k :
  fib n == 2 * (fib k) = ((k==0) && (n==0)) || ((n==3) && ((k==1) || (k==2))).
Lemma fib_times_c_is_fib4 n k :
  fib n == 3 * (fib k) = ((k==0) && (n==0)) || (((k==1) || (k==2)) &&(n==4)).
Lemma fib_times_c_is_fib5 n k m: 2 < k -> 2 < m ->
  fib n <> fib m * fib k.
```

The same technique says that $L_n = F_m L_k$ has no trivial solution. If $m = 1$ or $m = 2$, we are are reduced to the case $L_n = L_k$; which is $n = k$. If $m = 0$, we get $L_n = 0$, absurd, so we may assume $2 < m$. If $k = 1$, we get $L_n = F_m$, see above. The case $k = 0$ gives $L_n = 2F_m$, the only solution is $L_3 = 2F_3$. In the general case, we have $L_n = F_m L_{n-m+1} + F_{m-1} L_{n-m}$. This is less than the same quantity with m instead of $m-1$ and greater than the first term. Using the assumption and simplifying by F_m gives a contradiction.

Consider now $L_n = L_m L_k$. Without loss of generality, we assume $k \leq m$. If $k = 1$, we have the trivial solution $m = n$; if $k = 0$ we have $L_k = F_3$, and the previous results gives $L_3 = L_2 L_0$. Using (1.9) shows that there is no other solution.

Consider now $L_n = F_m F_k$. Without loss of generality, we assume $k \leq m$. We have $L_0 = F_1 F_3 = F_2 F_3$ and $L_3 = F_3 F_3$. If $k = 1$ or $k = 2$, we get $L_n = F_m$, see above. We use here (1.10), so that $5L_n = L_{m+k} \pm L_{m+k}$. In the general case, we may assume $k \geq 4$, so that $L_{m+k} = 5L_{m+k-4} + 3L_{m+k-5}$. Since $L_{m+k} \leq L_{m+k-5} < 2L_{m+k-5}$ it follows $n \leq m + k - 4$. This leads to a contradiction.

```
Lemma fib_times_lucas_is_lucas n k m:
  lucas n == (fib m) * (lucas k) =
  [ || ((m==1) || (m==2)) && (n == k),
    ((k==1) &&
    [|| (n==0) &&(m==3), (n==1) &&((m==1) || (m==2))  | (n==2)&&(m==4) ])
  | [&& k ==0 , n == 3 & m ==3] ].
Lemma lucas_times_lucas_is_lucas m n k: k <= m ->
  lucas n == (lucas m) * (lucas k) =
  [ || [&& k ==0, m == 0 & n == 3],
       [&& k ==0, m == 1 & n == 0] |
       [&& k ==1 & m == n] ].
Lemma fib_times_fib_is_lucas m n k: k <= m ->
  lucas n == (fib m) * (fib k) =
  [|| (m==3) && [|| (k==1) && (n==0), (k==2) && (n==0) | (k==3) && (n==3)],
    (k==1) && [|| (n==1)&&(m==1), (n==1) &&(m==2) | (n==2) && (m==4) ] |
    (k==2) && [|| (n==1) &&(m==2) | (n==2) && (m==4) ] ].
```

Consider now $F_m^2 + F_n^2 = F_k^2$ or $L_m^2 + L_n^2 = L_k^2$. By symmetry, we may assume $m \leq n$. In the Fibonacci case, the case $n = 0$ gives $m = k$; let's exclude this case. The relation $m \leq n$ implies $F_k^2 \leq 2F_n^2 < (9/4)F_n^2$, so that $F_k < (3/2)F_n \leq F_{n+1}$ On the other hand $n < k$ is obvious.

Consider now $L_m^2 + L_n^2 = F_k^2$. This case is not as easy as the previous one. There is the solution $L_2^2 + L_3^2 = F_5^2$, that reduces to $3^2 + 4^4 = 5^2$. By case analysis, we may assume $k > 5$, and by symmetry $m < n$. Replace k by $k+6$. Use (1.9) and (1.10). In each case there is a ± 2. We can get rid of these quantities by setting $x = 5(L_{2m} + L_{2n})$ and $y = L_{2k+12}$. We have then $x \leq y + 22$ and $y \leq x + 22$. Use $L_{a+5} = 5L_{a+1} + 3L_a$ with $a = 2k + 7$; it says $y + 22 < 5L_{2k+9}$. From $5L_{2n} \leq x$ it follows $2n < 2k + 9$. Since $m < n$ it follows $x \leq 5L_{2n+1}$, so that $2n = 2k + 8$, and $n = k + 4$. If we insert n in the previous inequalities and simplify we get $5L_{2m}$ is $6L_{2k+5} + 3L_{2k+4}$ plus or minus something ≤ 22. Since $22 < L_{2k+5} + 3L_{2k+4}$, it follows $2m > 2k + 5$, thus $m \geq k + 3$. The relation $y \leq x + 22$ leads to $L_{2k+2} \leq 22$, thus $k \leq 2$. Note that, if m and n have different parities, the $\pm$ cancel, and we can replace 22 by 2, and obtain a contradiction. Our proof is the following: if $m > k+3$, a contradiction is easy to obtain. Otherwise we have $m = k+3$ and $n = k+4$, $L_{2k+2} \leq 22$, and it suffices to check that the equation does not hold for k equal to 0, 1 or 2.

Consider now $F_m^2 + L_n^2 = F_k^2$. There is the solution $F_4^2 + L_3^2 = F_5^2$ (this is $3^2 + 4^4 = 5^2$ again). If $m = 0$, we are reduced to $L_n = F_2$, that has some solutions. There is no other solution for $n < 4$

(consider $x^2 + a^2 = y^2$ for $a = 1$, $a = 2$, $a = 4$; in each case, we give the list of all solutions). If $L_n \leq F_m$ we get $F_m^2 \leq F_k \leq 2F_m^2$, and there is no solution. One deduces that there is no solution for $k > n+2$; but obviously $k < n+2$ leads to a contradiction. So $k = n+2$. Replace n by $n+4$, and rewrite $L_{n+4} = F_{n+4} + 2F_{n+3}$ and simplify; we get $F_m^2 + 3F_{n+3}^2 = 3F_{n+4}^2$. The RHS is $< F_{n+6}^2$, so that $m < n+6$. Now $3F_{n+3}^2 < 2F_{n+4}^2$, so that $m > n+4$. The equation is equivalent to $F_m^2 = 3F_{n+2}F_{n+5}$. Using $m = n+5$ gives $F_{n+5} = 3F_{n+2}$. This gives $n = -1$ (thus the non-trivial solution).

```
Lemma fib_sum_square_is_fib_square m n k:
   0 < m <= n -> (fib m)^2  + (fib n)^2 = (fib k)^2 -> False.
Lemma fib_sum_square_is_fib_square_bis m n k:
  (fib m)^2  + (fib n)^2 == (fib k)^2 =
    ((m==0) && (fib n== fib k)) || ((n == 0) && (fib m== fib k)).
Lemma lucas_sum_square_is_lucas_square m n k:
   m <= n -> (lucas m)^2  + (lucas n)^2 = (lucas k)^2 -> False.
Lemma lucas_lucas_fib_square m n k: m <= n ->
  ((lucas m)^2 + (lucas n^2) == (fib k)^2) = [&& m==2, n==3 & k==5].
Lemma fib_sum_square_is_fib_square_bis m n k:
  (fib m)^2  + (fib n)^2 == (fib k)^2 =
    ((m==0) && (fib n== fib k)) || ((n == 0) && (fib m== fib k)).
Lemma lucas_fib_lucas_square m n k:
  ((fib m)^2 + (lucas n^2) == (fib k)^2) =
    ((m==0) && (fib k == lucas n)) || [&& k ==5, m==4 & n==3].
```

Chapter 2

Zeckendorf representations

Let's consider the following question: write an integer n as a sum of Fibonacci numbers: $n = \sum_{k\in I} F_k$. Each Fibonacci number should appear only once; as $F_1 = F_2$, we exclude the case $k = 1$. Moreover, since $F_0 = 0$, we will also exclude $k = 0$. For this reason, we define $Z_l = \sum_{k\in l} F_{k+2}$. Let $R(n)$ be the number of decompositions. We have $R(n) = 1$ for $n = 0$, $n = 1$, $n = 2$. More generally $R(n)$ is non-zero. We have $3 = F_4 = F_2 + F_3$, so that in general the decomposition is non-unique. We have $4 = F_4 + F_2$. Let $\tilde{N}(j)$ be the j-th number that can be written using only even indices. These numbers are 0, F_2, F_4, $F_2 + F_4$, F_6 etc. Note that $F_6 = F_5 + F_4 = F_5 + F_3 + F_2$; so that $R(F_6) = 3$. It happens that $R(\tilde{N}(j)) = s_{j+1}$, where s is the Stern sequence. This property is the link with the next chapter, but is not proven here.

We start with some results from [5].

2.1 Sums of Fibonacci numbers

A list $(l_1, l_2, \dots l_k, l_{k+1})$ is sorted by $\prec$ when $l_i \prec l_{i+1}$ for $0 \le i \le k$. Any list that has zero or one element is sorted. In the non-trivial case, '`sorted c (a::l)`' is expressed as '`path c a l`'. If $\prec$ is transitive, l is sorted by $\prec$, a is before b in l, then $a \prec b$. If moreover $\prec$ is irreflexive, then $a \neq b$, so that the list is `uniq` (no element appears more than once in the list).

Let's write $x \ll y$ or $y \gg x$ when $x + 2 \le y$. If a list is sorted by $\gg$, it is sorted by $>$, the converse being true if all elements of the list are even. If l is sorted by $\ll$ or $\gg$, so is the list of successors of l, the converse being true. As $\ll$ is transitive and irreflexive, a list sorted by $\ll$ is `uniq`; if a is the head of the list, all other elements are $\ge a + 2$, in particular are ≥ 2. The lists $(2a, 2a + 2, 2a + 4, \dots)$ and $(2a + 1, 2a + 3, 2a + 5, \dots)$ are sorted by $\ll$.

Note: if a list is `uniq`, one can sort it, and obtain a list sorted by $<$ or $>$.

```
Definition ggen := [rel m n | m >= n.+2].
Definition llen := [rel n m | ggen m n]
Definition even := [pred n | ~~ (odd n) ].

Lemma sorted_gtn l: uniq l -> sorted gtn (rev (sort leq l)).
Lemma transitive_llen:  transitive llen.
Lemma irreflexive_llen:  irreflexive llen.
Lemma sorted_ggenW s : sorted ggen s -> sorted gtn s.
Lemma sorted_ggenS s : all even s -> sorted gtn s -> sorted ggen s.

Lemma sorted_llen_uniq l: sorted llen l -> uniq l.
```

```
Lemma sorted_ggen_uniq l : sorted ggen l -> uniq l.
Lemma uniq_llen a l: sorted llen (a :: l) -> uniq l.
Lemma path_all_llen a l: path llen a l -> all (llen a) l.
Lemma sorted_llen_succ l: sorted llen (succ_seq l) = sorted llen l.
Lemma sorted_ggen_succ l: sorted ggen (succ_seq l) = sorted ggen l.

Lemma mkseq_uniq (f: nat -> nat) a n:
  injective f -> uniq [seq (f i) | i <- iota a n].
Lemma sorted_llen_mkseq f a n: (forall u, llen (f u) (f u.+1)) ->
  sorted llen [seq (f i) | i <- iota a n].

Lemma sorted_llen_mkseq_e a n: sorted llen [seq i.*2 | i <- iota a n].
Lemma sorted_ggen_sdouble n: sorted ggen (rev (mkseq double n)).
Lemma sorted_ge2_all_ps l: sorted ggen (rcons l 0) ->
   all (leq 2) l /\   all (leq 1) l.
```

If l is a list of integers, we define

$$Z_l = \sum_{k\in l} F_{k+2}, \qquad Z'_l = \sum_{k\in l} F_{k+1}, yy \qquad Z''_l = \sum_{k\in l} F_k.$$

The Fibonacci recurrence relation is then: $Z_l = Z'_l + Z''_l$. If $n = Z_l$, and l is sorted by $>$, then l is called a *representation* of n; if l is sorted by $\gg$, then l is called a *canonical representation* of n. If n is the greatest element of l, the relation $\sum_{i\le n} F_i = F_{n+2} - 1$ shows $Z_l \le F_{n+2} - 2$. If moreover the list is sorted by $x \ll y$ we have $Z_l < F_{n+1}$. This shows that any integer has at most one canonical representation.

```
Definition Zeck_val l := \sum_(i <-l) fib(i.+2).
Definition Zeck_valp l := \sum_(i <-l) fib(i.+1).
Definition Zeck_valpp l := \sum_(i <-l) fib i.


Lemma Zeckv_nil : Zeck_val nil = 0.
Lemma Zeckv_cons n l: Zeck_val (n ::l) = fib (n.+2) + Zeck_val l.
Lemma Zeckv_rev l: Zeck_val (rev l) = Zeck_val l.
Lemma Zeckvp_rev l: Zeck_valp (rev l) = Zeck_valp l.
Lemma Zeckv_pos n l: 0 < Zeck_val (n ::l).
Lemma Zeck_valppE l: Zeck_val l = Zeck_valp l +  Zeck_valpp l.

Lemma Zeckv_bound0 n l : path gtn n l -> (Zeck_val (n::l)).+2 <= fib (n.+4).
Lemma Zeckv_bound1 n l : path gtn n l -> Zeck_val (n::l) < fib (n.+4).
Lemma Zeckv_bound2 n l : path ggen n l -> Zeck_val (n::l) < fib (n.+3).
Lemma Zeckv_bound3 l n: uniq l -> (all (leq^~ n) l) ->
   (Zeck_val l).+2 <= fib n.+4.
Lemma Zeck2_unique (s s': seq nat) : sorted ggen s -> sorted ggen s' ->
   Zeck_val s = Zeck_val s' ->  s = s'.
```

Define $z(l)$ to be the triple $(Z_l, Z'_l, p(l))$, where $p(l)$ is 0 if the last element of l is even, 1 if it is odd, and 2 if the list is empty. If l is sorted by $>$, there is l' sorted by $\gg$, has the same z-value but is shorter (unless l is sorted by $\gg$). Proof. Let $A(l)$ be defined by induction as follows: it is l when l has less than two elements; if $l = a+1 :: a :: l'$, then $A(l) = a+2 :: A(l')$; if $l = b :: a :: l'$ where $b > a+1$ and $A(a :: l') = c :: l''$, then $A(l)$ is $c+2 :: l''$ when $b = c+1$, and $b :: c :: l''$ otherwise. Note that $z(a+1 :: a :: l) = z(a+2 :: l)$; moreover $z(l) = z(l')$ implies $z(a :: l) = z(a :: l')$. These two facts ensure that $z(A(l)) = z(l)$. Moreover, it follows by induction that $c \le a+1$ (and in the first case, that if l is not empty, the head of $A(l)$ is at most a). This implies that $A(l)$ is sorted by $\gg$.

Consequence: if l and l' are sorted by > and have the same Z-value they have the same Z′-value and p-value (since $\mathrm{A}l(l) = \mathrm{A}(l')$). This could be weakened to: if two lists are unique with the same Z-value, then they have the same Z′-value. If $n = \mathrm{Z}_l$ then $p(l)$ depends only on n, not on l. We shall say that n is of type even or odd accordingly (we shall see below that each integer has a representation).

```
Definition odd_last l := if l is a::b then (odd(last a b)): nat else 2.

Lemma Zeck_valp_aux l: sorted gtn l ->
  exists (l': seq nat),
  [/\ sorted ggen l', odd_last l = odd_last l',
    Zeck_val l = Zeck_val l', Zeck_valp l = Zeck_valp l' &
     size l' <= size l ?= iff (sorted ggen l)].
Lemma Zeck_valp_same (l1 l2: seq nat): sorted gtn l1 -> sorted gtn l2 ->
    Zeck_val l1 =  Zeck_val l2 ->
    Zeck_valp l1 =  Zeck_valp l2 /\  odd_last l1 = odd_last l2.
```

2.2 Existence of a representation

We define the *fib-content* of n as the greatest k such that $\mathrm{F}_k \le n$. If $n > 0$, this index is at least two, so we shift n by one and k by two, and consider $c(n)$ satisfying:

$$\mathrm{F}_{c(n)+2} \le n+1 < \mathrm{F}_{c(n)+3}.$$

The second inequality says $n+1-\mathrm{F}_{c(n)+2} < \mathrm{F}_{c(n)+1}$. So, its fib-content is $< c(n-1)$. Note that $c(\mathrm{F}_{n+2}-1) = n$.

```
Fact fib_cont_aux1 n: exists i, (fib i.+2) <= n.+1.
Fact fib_cont_aux2 n i: (fib i.+2) <= n.+1 -> i <= n.

Definition fib_content n := ex_maxn (fib_cont_aux1 n) (@fib_cont_aux2 n).
Lemma fib_contentP n (r := fib_content n):  fib (r.+2) <= n.+1 < fib (r.+3).
Lemma fib_content_large m: m.+1 - fib (fib_content m).+2 <= m.
Lemma fib_content_lt n (r := fib_content n): n.+1 - fib (r.+2) < fib r.+1.
Lemma fib_content_eq n: fib_content ((fib n.+2).-1) = n.
```

2.2.1 Minimal representations

We define inductively a function by $\mathrm{Zeck}(n+1) = c(n) :: \mathrm{Zeck}(n+1-\mathrm{F}_{c(n)+2})$; the value of $\mathrm{Zeck}(0)$ being the empty list.

```
Fixpoint Zeck_rec n k:=
   if k is k'.+1 then
     if n is n'.+1 then let r := fib_content n' in
       r:: (Zeck_rec (n - (fib (r.+2))) k')
    else nil
   else nil.
Definition Zeck n := Zeck_rec n n.
Lemma Zeck_0 : Zeck 0 = nil.
Lemma Zeck_S' n (r := fib_content n) :
   Zeck n.+1 = r :: (Zeck (n.+1 - (fib (r.+2)))).
```

If $F_{k+2} \le n < F_{k+3}$ then $\mathrm{Zeck}(n) = k :: \mathrm{Zeck}(n - F_{k+2})$. The value $\mathrm{Zeck}(n)$ is a representation of n, called the *canonical representation* of n. It is the unique representation sorted by $\gg$; it is also the representation whose length is minimal. The following property will be used: if $n \neq 0$, there is a non-empty list s, sorted by $\ll$ such that $n = Z_s$ and $\mathrm{Zeck}(n)$ is the reverse of s.

```
Lemma Zeck_S k n: fib k.+2 <= n < fib k.+3 ->
  Zeck n = k :: Zeck (n - fib k.+2).
Lemma Zeck_fib n: Zeck (fib n.+2) = [:: n].
Lemma Zeck_1 : Zeck 1 = [:: 0].
Lemma Zeck_2 : Zeck 2 = [:: 1].
Lemma Zeck_ggen n: sorted ggen (Zeck n).
Lemma Zeck_uniq n: uniq (Zeck n).
Lemma Zeck_zeck_val n: Zeck_val (Zeck n) = n.
Lemma ZeckP l: sorted ggen l -> Zeck (Zeck_val l) = l.
Lemma Zeck_prop1 n: n != 0 -> exists a l,
  [/\ sorted llen (a::l), Zeck n = rev (a::l) & n = Zeck_val (a::l) ].
Lemma Zeck_is_minimal l n (L := Zeck n):
   sorted gtn l -> Zeck_val l = n -> size L <= size l ?= iff (l == L).
```

Let's define $\mu(n)$ to be the least element of $\mathrm{Zeck}(n)$. If n is non-zero, this is $< n$, so that we can take n as default value when the list is empty. If $k < \mu(n)$, then all elements i of $\mathrm{Zeck}(n)$ satisfy $k < i$ (in particular are non-zero).

```
Definition Zeck_li n := last n (Zeck n).

Lemma Zeck_val_bounded n: all (fun i => fib i.+2 <= n) (Zeck n).
Lemma Zeck_val_bounded1 n: all (fun i => i < n) (Zeck n).
Lemma Zeck_li_pr n: n != 0 ->  Zeck_li n < n.
Lemma Zeck_li_prop1 n : all (leq (Zeck_li n)) (Zeck n).
Lemma Zeck_li_prop2 n k:
   k < (Zeck_li n) -> all [pred z | k < z] (Zeck n).
```

We define $e(n)$ to be Z'_l where $l = \mathrm{Zeck}(n)$. Note that any another list l can be used, provided that $Z_l = n$ and l has no duplicates. The Fibonacci recurrence says $n = e(n) + Z''_l$, and, if s is the list shifted by one, so that $Z_l = Z'_s$, $Z'_l = Z''_s$ and $Z_l + Z'_l = Z_s$ (this holds whatever l). We also have: $e(n) \le n$, and equality holds if and only if $n \le 1$; $e(Z'_l) = Z''_l$ provided that l has no duplicates and does not contain zero; the last elements of l and $\mathrm{Zeck}(Z_l)$ have the same parity. Note that $e(x + Z_l) = e(x) + e(Z_l)$ provided that for some a with $x \le F_{a+2}$, the list $a :: l$ is sorted by $\ll$.

```
Definition Zeckp n := Zeck_valp (Zeck n).

Lemma Zeckp_0: Zeckp 0 = 0.
Lemma Zeckp_1: Zeckp 1 = 1.
Lemma Zeckp_eq0 n: (Zeckp n == 0) = (n == 0).
Lemma Zeckp_prop00 l: sorted gtn l ->
  Zeckp (Zeck_val l) =  Zeck_valp l /\
  odd_last(Zeck (Zeck_val l)) =  odd_last l.

Lemma Zeckp_prop0 l: sorted gtn l -> Zeckp (Zeck_val l) =  Zeck_valp l.
Lemma Zeckp_prop1 (l: seq nat): uniq l -> Zeckp (Zeck_val l) = Zeck_valp l.
Lemma Zeckp_prop2 n : n = Zeckp n + Zeck_valpp (Zeck n).
Lemma Zeckp_prop1_bis l:  all (leq 1) l -> uniq l ->
  Zeckp (Zeck_valp l) =  Zeck_valpp l.
```

```
Lemma Zeckp_prop2_bis l (s := [seq i.+1 | i <- l]):
  [/\ Zeck_valp s = Zeck_val l,  Zeck_valpp s = Zeck_valp l &
     Zeck_val l + Zeck_valp l = Zeck_val s].
Lemma Zeck_split a l x: sorted llen (a :: l) -> x <= fib a.+2 ->
    Zeckp (x + (Zeck_val l)) = Zeckp x + Zeckp (Zeck_val l).
Lemma Zeckp_le n: Zeckp n <= n ?= iff (n <= 1).
Lemma Zeck_odd_last_prop l: sorted gtn l ->
  odd_last(Zeck (Zeck_val l)) =  odd_last l.
```

Let L be the list of the k first even integers. The previous lemmas about sums of even/odd -based Fibonacci numbers can be expressed as $Z''_L = F_{2k-1} - 1$ and $Z'_L = F_{2k}$. The first formula is $Z_L = F_{2k+1} - 1$, since $F_0 = 0$. It follows $e(F_{2n+1} - 1) = F_{2n}$. Write L as $0 :: L'$. Then $F_{2n+1} - 2 = Z_{L'}$ and $F_{2n} - 1 = Z'_{L'}$; it follows $e(F_{2n+1} - 2) = F_{2n} - 1$. The two quantities $e(F_{2n+2} - 1)$ and $e(F_{2n+2} - 2)$ are equal to $F_{2n+1} - 1$. The expression is $e(x + s)$ where x is respectively F_3 or F_1, and s the sum of odd-based Fibonacci numbers (starting at F_5). It is $e(x) + e(s)$, but $e(x) = F_2$, this the sum of even-based Fibonacci.

$$e(F_{2n+1} - 1) = F_{2n}, \quad e(F_{2n+1} - 2) = F_{2n} - 1, \quad e(F_{2n} - 1) = e(F_{2n} - 2) = F_{2n-1} - 1.$$

```
Lemma fib_sum_even_alt n: Zeck_valpp (mkseq double n) = (fib (n.*2.-1)).-1.
Lemma fib_sum_odd_alt n: Zeck_valp (mkseq double n) = fib (n.*2).
Lemma fib_sum_even_alt2 n: Zeck_val (mkseq double n) = (fib (n.*2.+1)).-1.

Lemma Zeckp_fib n: Zeckp (fib n.+2) = fib n.+1.
Lemma Zeck_fib1 n: Zeck (fib n.*2.+1).-1 = rev (mkseq double n).
Lemma Zeck_fib2 n: Zeck (fib n.*2.+2).-1 = rev (succ_seq (mkseq double n)).

Lemma Zeckp_fiba n: Zeckp (fib (n.*2.+1)).-1 = fib n.*2.
Lemma Zeckp_fibb n: Zeckp (fib n.*2.+1).-2 = (fib n.*2).-1.
Lemma Zeckp_fibc n: Zeckp (fib n.*2).-1 = (fib n.*2.-1).-1.
Lemma Zeckp_fibd n: Zeckp (fib n.*2).-2 = (fib n.*2.-1).-1.
```

Assume $0 \notin \mathrm{Zeck}(n)$, $\mu(n')$ odd, m arbitrary. Then

$$e(e(n)) = n - e(n), \qquad e(e(n') - 1) = (n' - e(n')) - 1, \tag{2.1}$$
$$m = e(m + e(m)) = e(m + e(m - 1)).$$

The first relation comes from $n = e(n) + Z''_l$, where $l = \mathrm{Zeck}(n)$. For the second, write $n' = p + F_{2k+3}$. We have $e(n') = e(p) + e(F_{2k+3}) = e(p) + F_{2k+2}$, so that $e(e(n') - 1) = e(e(p)) + e(F_{2k+2} - 1) = e(e(p)) + F_{2k+1} - 1$. Now $e(e(p)) = p - e(p)$, and $F_{2k+1} = F_{2k+3} - F_{2k+2}$. The last two formulas are shown in the same way.

```
Lemma Zeckp_prop3a n (e := Zeckp):  e (n + e n) = n.
Lemma Zeckp_prop3b n (e := Zeckp):  Zeck_li n != 0 ->  e (e n) = n - e n.
Lemma Zeckp_prop4a n (e := Zeckp):  e (n + e n.-1) = n.
Lemma Zeckp_prop4b n (e := Zeckp): odd (Zeck_li n) ->
   (e (e n).-1).+1 = n - e n.
```

2.2.2 Maximal representations

We have shown above an algorithm that produces a minimal representation from an arbitrary one. What happens if we proceed in the other way? The result cannot have big holes: if a is followed by b then $a = b + 1$ or $a = b + 2$ for otherwise a could be replaced by the pair

$a+1, a$. Moreover, the last element c must be 0 or 1, otherwise it could be replaced by the pair $c-1, c-2$.

The first condition can be expressed by: the list is sorted by R, where $R(a,b)$ is $a = b+1 \vee a = b+2$; we say that the list satisfies (R) if moreover the second condition holds. There are different ways to express it, for instance as $R(c+1,0)$ or as $c \leq 1$.

```
Definition gespec := [rel i j | (i == j.+1) || (i== j.+2) ].
Definition lespec := [rel i j | gespec j i].
Definition spec_sorted l := sorted gespec l && gespec (last 0 l).+1 0.

Lemma leqn1 n: (n <=1 ) = (n==0) || (n ==1).
Lemma spec_sorted_rev l:
  (spec_sorted (rev l)) = sorted lespec l && lespec 0 (head 0 l).+1.
Lemma spec_sorted_rcons l i:
  (spec_sorted (rcons l i)) = (sorted lespec (i:: rev l)) && (i <= 1).
Lemma spec_sorted_sorted l: spec_sorted l -> sorted gtn l.
Lemma spec_sorted_nil: spec_sorted [::].
Lemma spec_sorted_rec a l: spec_sorted (a::l) =
  (if l== nil then a <=1  else gespec a (head 0 l)) && spec_sorted l.
```

We say that l is a *maximal representation* of n if it satisfies (R) and $Z_l = n$. Example: the canonical representation of $F_n - 1$ is maximal. Consider the relation

$$F_k + F_{k+1} + F_{k+3} + \ldots + F_{k+2i+1} = F_{k+2i+2}.$$

The sequence of indices (in decreasing order) is ordered by R. Thus, this is the maximal representation of F_n ($n \geq 4$) when $k = 2$ or $k = 3$ (for $n \leq 3$ the maximal representation is trivial). We deduce: every representation can be converted to a maximal one; proof by induction. Consider a list $a :: l$, in decreasing order. If l is empty, the previous formula gives a maximal representation. Otherwise, apply the induction assumption to l, this gives a list $b :: l'$. Note that b is not greater than the head of l, so $a > b$. If $a = b+1$ or $a = b+2$, the result is $a :: b :: s$. Otherwise choose i such that $a = k+2i+2$, $k = b+1$ or $k = b+2$. The result is then $k+2i+1, \ldots, k+3, k+1, k, b :: s$.

```
Definition max_rep n l:= spec_sorted l && (Zeck_val l == n).

Lemma max_rep_fib1 n: max_rep (fib n.*2.+1).-1  (rev (mkseq double n)).
Lemma max_rep_fib2 n:
  max_rep (fib n.*2.+2).-1 (rev (succ_seq (mkseq double n))).
Lemma max_rep_fib3 n:
  max_rep (fib n.*2.+2) (rev (0::(succ_seq (mkseq double n)))).
Lemma max_rep_fib4 n:
  max_rep (fib n.*2.+3) (rev (1:: (map double (iota 1 n)))).
Lemma fibp_aux i k:  k + (i.+1).*2 = k.+2 + i.*2.
Lemma fib_partial_sum k n (L := k:: mkseq (fun i => (k.+1+ i.*2)) n) :
   \sum_(i <- L) (fib i) = fib (k + n.*2) /\ sorted gespec (rev L).
Lemma fib_partial_sum' k n (L := k:: mkseq (fun i => (k.+1+ i.*2)) n) :
  Zeck_val L = fib (k.+2 + n.*2) /\ sorted gespec (rev L).
Lemma max_rep_prop1 (l: seq nat): sorted gtn l ->
 { l' | [/\ spec_sorted l', Zeck_val l = Zeck_val l' &
    size l <= size l' ?= iff (spec_sorted l)] }.
```

We denote by $\text{Zeck}_M(n)$ the maximal representation obtained by applying the previous algorithm to $\text{Zeck}(n)$. If l is a list that satisfies (R) and starts with a then

$$F_{a+3} - 1 \leq Z_l \leq F_{a+4} - 2, \tag{2.2}$$

so that a (thus the list l) is uniquely determined by the value Z_l. So we can state: every integer has a unique maximal representation, and its size is maximal. Combining with preceding results, we get: if l is any representation, $s(\text{Zeck}(n)) \le s(l) \le s(\text{Zeck}_M(n))$; moreover, if $\text{Zeck}(n)$ and $\text{Zeck}_M(n)$ have the same size, then $l = \text{Zeck}(n)$.

It follows: n has a unique representation if and only if n is the predecessor of a Fibonacci number. In fact, n has a unique representation if and only if $\text{Zeck}(n) = \text{Zeck}_M(n)$. This condition is true if n is the predecessor of a Fibonacci number. Conversely, if it holds, then $\text{Zeck}(n)$ is a list sorted by $\gg$ satisfying R. If the two first elements of the list are a and b, then $a \ge b+2$ on one hand, and is $b+1$ or $b+2$, on the other hand, thus $a = b+2$. So the list has the form $a, a-2, a-4\ldots$. Moreover, the last element is either 0 or 1, thus the result.

```
Definition ZeckM n := sval (max_rep_prop1 (sorted_ggenW (Zeck_ggen n))).

Lemma fib_ge2_alt n: fib n.+3 = (fib n.+3).-2.+2.
Lemma max_rep_bound a l: spec_sorted (a::l) ->
   (fib a.+3).-1 <= Zeck_val (a::l) < (fib a.+4).-1.
Lemma max_rep_unique (s s': seq nat) : spec_sorted s -> spec_sorted s' ->
   Zeck_val s = Zeck_val s' ->  s = s'.
Lemma ZeckM_prop1 n:
  [/\ spec_sorted (ZeckM n), Zeck_val (Zeck n) = Zeck_val (ZeckM n)
    & size (Zeck n) <= size (ZeckM n) ?= iff spec_sorted (Zeck n)].
Lemma ZeckM_prop2 n: max_rep n (ZeckM n).
Lemma ZeckM_prop3 n l: max_rep n l -> ZeckM n = l.
Lemma ZeckM_is_maximal n l: sorted gtn l -> Zeck_val l = n ->
    size l <= size (ZeckM n) ?= iff (l == (ZeckM n)).
Lemma ZeckM_prop4 n l: sorted gtn l -> Zeck_val l = n ->
  (size (Zeck n) <= size l <= size (ZeckM n) /\
  (size (Zeck n) = size (ZeckM n) -> l = (Zeck n))).
Lemma unique_representation n:
  (exists k, n = (fib k.+2).-1) <->
   (forall l l', sorted gtn l -> Zeck_val l = n ->
     sorted gtn l' -> Zeck_val l' = n -> l = l').
```

We restate (2.2) as: if a is the head of $\text{Zeck}_M(n)$, then $F_{a+3} \le n+1 < F_{a+4}$, and conversely. One deduces that $\text{Zeck}_M(n+F_{a+3}) = a+1 :: \text{Zeck}_M(n)$ and $\text{Zeck}_M(n+F_{a+4}) = a+2 :: \text{Zeck}_M(n)$. If $n = 0$, these formulas hold for $a = -1$; for this reason, we give the value of $\text{Zeck}_M(n)$ for $n \le 2$; in all other cases, the list has at least two elements.

```
Lemma ZeckM_bound1 n (a := head 0 (ZeckM n)):
  0 < n -> (fib a.+3).-1 <= n < (fib a.+4).-1.
Lemma ZeckM_bound2 n a:
  (fib a.+3).-1 <= n < (fib a.+4).-1  -> a = head 0 (ZeckM n).

Lemma ZeckM0: ZeckM 0 = [::].
Lemma ZeckM1: ZeckM 1 = [:: 0].
Lemma ZeckM2: ZeckM 2 = [:: 1].
Lemma ZeckMgt2 n: 2 <n -> 2 <= size (ZeckM n).
Lemma ZeckM_rec1 n a: (fib a.+3).-1 <= n < (fib a.+4).-1 ->
  ZeckM (n + (fib a.+3)) = a.+1::ZeckM n.
Lemma ZeckM_rec2 n a: (fib a.+3).-1 <= n < (fib a.+4).-1 ->
  ZeckM (n + (fib a.+4)) = a.+2::ZeckM n.
```

Consider a sorted list l whose greatest element is a. There are at most $a+1$ elements in the list. If there are $a+1$ elements, then the list contains all integers $\le a$; this means $Z_l = F_{a+4} - 2$.

We restate this as: if $F_{a+3} \le n+1 < F_{a+4}$, then $\mathrm{Zeck}_M(n)$ has at most $a+1$ elements, and exactly this number when $n = F_{a+4} - 2$.

```
Lemma ZeckM_fibm2 n:  ZeckM (fib (n.+3)).-2 = rev (iota 0 n).
Lemma size_sorted a l:
  spec_sorted (a::l) -> size l <= a ?= iff (l == rev (iota 0 a)).
Lemma ZeckM_bound3 n a:
  (fib a.+3).-1 <= n < (fib a.+4).-1 ->
  size (ZeckM n) <= a.+1 ?= iff (n== (fib (a.+4)).-2).
```

Let $\mathcal{V}_{n,m}$ be the set of integers i in the interval $J_n = [F_n - 1, F_{n+1} - 1[$ such that $\mathrm{Zeck}_M(i)$ is of length m, and let $V_{n,m}$ be its cardinal. We pretend that

$$V_{n,m} = \binom{m}{n-m-2}.$$

The sets J_n are empty for $n \le 2$, and form a partition of the integers otherwise; since $J_2 = \{0\}$ it follows $V_{0,m} = V_{1,m} = 0$ and $V_{2,m} = \delta_{m0}$. We also have $V_{n,0} = \delta_{n2}$ (if $i \in \mathcal{V}_{n,0}$, then $\mathrm{Zeck}_M(i)$ is empty, $i = 0$, and $n = 2$).

We have seen above that J_{n+3} is the set of all integers i such that $l = \mathrm{Zeck}_M(i)$ starts with n; in particular l has at most $n+1$ elements (this shows $V_{n,m} = 0$ when $n < m+2$) and if l has $n+1$ elements then i is the greatest element of J_{n+3}. This says $V_{n+2,n} = 1$. Our objective becomes $V_{n+m+2,m} = \binom{m}{n}$, and it suffices to prove that V satisfies the same recurrence relation than the binomial coefficient, namely $V_{n+2,m+1} = V_{n+1,m} + V_{n,m}$ for $n \ge m+2$. We use

$$\sum_{a+d \le i < b+d} f(i) = \sum_{a \le i < c} f(i+d) + \sum_{c \le i < b} f(i+d)$$

valid when $a \le c \le b$. Here $a = F_n - 1$, $b = F_{n+2} - 1$, $c = F_{n+1} - 1$ and $d = F_{n+1}$. The first sum is over J_{n+2}, and the two other over J_n and j_{n+1}. The trick is now that if $j = i + F_{n+1}$, then $\mathrm{Zeck}_M(j)$ has one more element than $\mathrm{Zeck}_M(i)$, so that $f(i+d)$ (namley $\mathrm{Zeck}_M(j)$ has size $m+1$) becomes: $\mathrm{Zeck}_M(i)$ has size m.

```
Definition card_max_rep n m:=
   \sum_((fib n).-1 <= i < (fib n.+1).-1) (size (ZeckM i)== m).

Lemma card_max_repE n m:
  card_max_rep n m =
    \sum_((fib n).-1 <= i < (fib n.+1).-1 | (size (ZeckM i)== m)) 1.
Lemma card_max_rep0m m: card_max_rep 0 m = 0.
Lemma card_max_rep1m m: card_max_rep 1 m = 0.
Lemma card_max_rep2m m: card_max_rep 2 m = (m==0).
Lemma card_max_rep41: card_max_rep 4 1 = 1.
Lemma card_max_rep_smalln n m: n < m.+2 -> card_max_rep n m = 0.
Lemma card_max_repn0 n: card_max_rep n.+3 0 = 0.
Lemma card_max_repn2n n: card_max_rep n.+2 n = 1.
Lemma card_max_rep_rec n m: m.+2 <= n ->
  card_max_rep n.+2 m.+1 = card_max_rep n.+1 m + card_max_rep n m.
Lemma card_max_rep_val n m: m.+2 <= n -> card_max_rep n m = 'C(m,n- m.+2).
```

Let now $U_{n,m}$ be the number of integers i in J_n for which $f(i) = m$ where $f(i)$ is the length of $\mathrm{Zeck}(i)$, and $J_n = [F_n, F_{n+1}[$.

```
Definition card_min_rep n m:=
   \sum_((fib n) <= i < (fib n.+1)) (size (Zeck i)== m).

Lemma card_min_repE n m:
  card_min_rep n m =
    \sum_((fib n) <= i < (fib n.+1)| (size (Zeck i)== m)) 1.
```

We show

$$\mathrm{U}_{n,m} = \binom{n-m-1}{m-1}.$$

Note that $\mathrm{J}_0 = \{0\}$, $\mathrm{J}_1 = \emptyset$, $\mathrm{J}_2 = \{1\}$, all other intervals have at least two elements and these sets form a partition of the integers. The case of n small is trivial. Assume $m = 0$. This gives $i = 0$, thus $n = 0$. Assume $m = 1$. This means that i is a Fibonacci number. There is only one such number, the first of J_n (when $n \geq 2$).

```
Lemma card_min_rep0m m: card_min_rep 0 m = (m==0).
Lemma card_min_rep1m m: card_min_rep 1 m = 0.
Lemma card_min_rep2m m: card_min_rep 2 m = (m==1).
Lemma card_min_repn0 n: card_min_rep n 0 = (n==0).
Lemma card_min_repn1 n: card_min_rep n 1 = (2 <=n).
```

We have $\mathrm{U}_{n+2,m+1} = \mathrm{U}_{n+1,m+1} + \mathrm{U}_{n,m}$. Assume $i \in \mathrm{J}_{n+2}$. We have $\mathrm{Zeck}(i) = n :: \mathrm{Zeck}(i - \mathrm{F}_{n+2})$. Assume $\mathrm{F}_{n+2} + \mathrm{F}_n \leq i$. This means $j = i - \mathrm{F}_{n+2} \in \mathrm{J}_n$, and $\mathrm{Zeck}(i)$ has one more element than $\mathrm{Zeck}(j)$. Assume on the other hand that $i < \mathrm{F}_{n+2} + \mathrm{F}_n$, and let $k = i - \mathrm{F}_n$. We have $k \in \mathrm{J}_{n+1}$ so that $\mathrm{Zeck}(k) = n - 1 :: \mathrm{Zeck}(k - \mathrm{F}_{n+1})$. Now $k - \mathrm{F}_{n+1} = i - \mathrm{F}_n - \mathrm{F}_{n-1} = j$ so that $\mathrm{Zeck}(i)$ has the same length as $\mathrm{Zeck}(k)$. One deduces the main result. Note that $\mathrm{U}_{n,m} = 0$ when $0 < n \leq m$ by the length condition.

```
Lemma card_min_rep_rec n m:
  card_min_rep n.+2 m.+1 = card_min_rep n.+1 m.+1 + card_min_rep n m.
Lemma card_min_rep_small n m: 0 < n <= m -> card_min_rep n m = 0.
Lemma card_min_rep_val n m :  card_min_rep (n+m.+2) m.+1 = 'C(n,m).
Lemma card_min_rep_valbis n m: 0< m < n ->
   card_min_rep n m = 'C(n-m-1,m.-1).
```

2.2.3 A characterization of the Fibonacci numbers

According to [2], the only sequence v such that each integer n can be uniquely written as $n = \sum_{i \in l} v_i$, where l satisfies (R), is the Fibonacci sequence.

We state here some useful properties.

```
Lemma iota_lespec_sorted i n: sorted lespec (iota i n).
Lemma iota_rem_lespec_sorted i j n: sorted lespec (rem j (iota i n)).
Lemma iota_rem_spec_sorted j i: spec_sorted (rev (rem i (iota 0 j))).
Lemma iota_rem2_spec_sorted j i:
  spec_sorted (rev (rem i.+2 (rem i (iota 0 j)))).
```

Note that v is injective: assume $v_i = v_j$, let l be the list of all integers $\leq i + j$ in decreasing order, l_i and l_j the lists obtained from l by removing respectively i and j. The two sequences satisfy (R) and have the same value, so are equal. Now i is not in l_i but is in l_j when $i \neq j$, so that $i = j$. A similar argument says that v_i cannot be zero. In the same way, one cannot have $v_i = v_{i-1} + v_{i+1}$ nor $v_i + v_{i+2} = v_{i+1} + v_{i+3}$.

```
Section DualUniqueRepresentation.

Parameter v: nat -> nat.
Definition Zval_v l:= \sum_(i <- l) (v i).
Hypothesis v_exists:
  forall n, exists2 l, spec_sorted l & Zval_v l = n.
Hypothesis v_unique:
  forall l l', spec_sorted l ->  spec_sorted l' -> Zval_v l = Zval_v l' ->
   l = l'.

Lemma DUR_injv: injective v.
Lemma DUR_positive i: 0 < v i.
Lemma DUR_exclusion1 i: v i + v i.+2 != v i.+1.
Lemma DUR_exclusion2 i: v i + v i.+2 != v i.+1 + v i.+3.
```

We have $v_0 = 1$ and $v_1 = 2$. This is a bit convoluted. Consider the representation of 2; this is a list with one element, say b, with $b = 0$ or $b = 1$. Thus, one of v_0 and v_1 is 2; by considering the representation of 1, it follows that the other one is 1. It may happen that $v_0 = 2$ and $v_1 = 1$. Let's show that this case is excluded by considering representations of 4 and 5. First note that there are two terms a_4 and b_4 and a_5 and b_5 (if there are at least 3 terms, the sum is at least $1+2+3=6$), where b is 0 or 1, and $a = b+1$ or $b+2$. Assume $b_5 = 1$. Thus $v_{a_5} = 4$, where a_5 is 2 or 3; as the representation of 4 is $4 = 1+3$, the first terms of the sequence are 2, 1, 3, 4 or 2, 1, 4, 3; in each case, one of the exclusion rules shown above applies. Thus $b_5 = 0$ and $a_5 = 2$, this gives $v_2 = 3$. Consider now the representations of 7 or 8; there are two or three terms. An examination of all possibilities show that there cannot be three terms for 7. If the expansion of 7 or 8 has two terms, one gets 6 or 7 for v_3 respectively. It follows that $v_3 = 6$, and the expansion of 8 has three terms. All these cases are excluded (in one case, we get $v_4 = 4$, so that $v_1 + v_3 = v_2 + v_4$, absurd).

We prove $\forall n, v_n = \mathrm{F}_{n+2}$ by induction on n. If l is a list, we denote by T_l the sum $\sum_{i\in l} v_i$. If l is the list $(n+1, n-1, n-3, \ldots)$, l' the same list, without the first term, then $\mathrm{Z}_{l'} = \mathrm{F}_{n+2} - 1$, $\mathrm{Z}_l = \mathrm{F}_{n+4} - 1$, and $\mathrm{Z}_l = \mathrm{F}_{n+3} + \mathrm{Z}_{l'}$.

Claim 1: if $v_i = \mathrm{F}_{i+2}$ for $i \le n$ then $v_{n+1} < \mathrm{F}_{n+3}$ is aburd. Let l be as above. We have $\mathrm{T}_{l'} = \mathrm{Z}_{l'}$ by the first assumption. From $\mathrm{T}_l = v_{n+1} + \mathrm{T}_{l'}$ and the second assumption we deduce $\mathrm{T}_l < \mathrm{Z}_l$. This inequality says that $\mathrm{Zeck}_{\mathrm{M}}(\mathrm{T}_l)$ is a list L whose largest term is $\le n$. This implies $\mathrm{T}_{\mathrm{L}} = \mathrm{Z}_{\mathrm{L}}$ and means that T_l has two representations: l and L, which are obviously different; absurd.

Claim 2: if $v_i = \mathrm{F}_{i+2}$ for $i < n$ then $v_{n+1} < \mathrm{F}_{n+1}$ is aburd. Same notations as above. We still have $\mathrm{T}_{l'} = \mathrm{Z}_{l'}$ since n is not in l. So $\mathrm{T}_l < \mathrm{F}_{n+1} + \mathrm{F}_{n+2} - 1$. So, largest term of L is $< n$, $\mathrm{T}_{\mathrm{L}} = \mathrm{Z}_{\mathrm{L}}$ and the conclusion is as above.

Claim 3: if $v_i = \mathrm{F}_{i+2}$ for $i \le n$ then $v_{n+1} > \mathrm{F}_{n+3}$ is absurd. Let $x = 1 + \sum_{i\le n} v_i$, and l its representation. There is at least one index in l which is $> n$. Assume $n+1 \in l$. Thus $x = y + v_{n+1} + \sum v_k$. Each v_k is F_{k+2}, so that $x > \sum v_k$ (where $n+1$ is an index in the sum). The sum is at least $\mathrm{F}_{n+4} - 1$, but this quantity is x. So l contains $n+2$, but not $n+1$. It thus contains n, so that $x \ge v_{n+2} + \mathrm{F}_{n+3} - 1$. From $x = \mathrm{F}_{n+4} - 1$, we deduce $v_{n+2} \le \mathrm{F}_{n+2}$. By injectivity of v it follows $v_{n+2} < \mathrm{F}_{n+2}$. The result follows by claim 2.

```
Section DualUniqueRepresentation.

Parameter v: nat -> nat.
Definition Zval_v l:= \sum_(i <- l) (v i).
Hypothesis v_exists:
```

```
  forall n, exists2 l, spec_sorted l & Zval_v l = n.
Hypothesis v_unique:
  forall l l', spec_sorted l ->  spec_sorted l' -> Zval_v l = Zval_v l' ->
   l = l'.

Lemma DUR_injv: injective v.
Lemma DUR_positive i: 0 < v i.
Lemma DUR_exclusion1 i: v i + v i.+2 != v i.+1.
Lemma DUR_exclusion2 i: v i + v i.+2 != v i.+1 + v i.+3.
Lemma DUR_01 : v 0 = 1 /\ v 1 = 2.
Lemma DUR_fib i : v i = fib i.+2.
End  DualUniqueRepresentation.
```

2.3 Counting the number of representations

We want to count the number of sequences l such that $Z_l = n$. This means that we compute the cardinal of some set E. As the set of all lists l satisfying some property is in general not finite, we proceed in an indirect way.

Let I_B be the type of all integers less than B. This type is empty for B = 0, so that we prefer to consider $I = I_{B+1}$. An element E of $\mathfrak{P}(I)$ is a set whose members are of type I. This can be converted to a list l, and each element of l can be converted to an integer. The result will be a list of integers, all ≤ B; the list has no duplicates, but the order of the terms is unspecified. Conversely, given a list l, one can consider its elements in I, and convert this to a set E. Obviously, if we start from a set, convert it to a list, then back to a set, we get the initial set. On the other hand, if we start from a list, assume all elements are ≤ B, and no duplicated, if we convert the list to a set, and the set to a list, we get a list with the same elements (maybe in a different order) that the orignal list. If E is a set, we define Z_E and Z'_E similarly to Z_l and Z'_l.

```
Definition seto_to_seq B (l: {set 'I_B.+1}) :=
  [seq (nat_of_ord i) | i <- enum l].
Definition seq_to_seto B l :=
  [set i | i in [seq (inord i : 'I_B.+1) | i <- l]].

Definition Zeck_sval B (t:{set 'I_B.+1}) := \sum_(i in t) (fib i.+2).
Definition Zeck_svalp B (t:{set 'I_B.+1}) := \sum_(i in t) (fib i.+1).

Lemma seto_to_seqK B (s: {set 'I_B.+1}) : seq_to_seto B (seto_to_seq s) = s.
Lemma seq_to_setoK B (l: seq nat):
  uniq l -> (all (fun z => z < B.+1) l) ->
  perm_eq (seto_to_seq (seq_to_seto B l)) l.
Lemma uniq_seto_to_seq B l: uniq (@seto_to_seq B l).
```

If E is any set, converted to a list l, then $Z_E = Z_l$ (this explains why we use the same notations). On the other hand, consider a list l, converted to a set E; assume that l has no duplicates; let $n = Z_l$, and $n \le B$. Then all elements of l are ≤ B (we can do better: if $i \in l$, then $F_{i+2} \le Z_l$). If we convert E to a list l', then l and l' have the same elements, thus the same Z value, so that $Z_E = n$. The same result holds when Z is replaced by Z′.

```
Lemma Zeckv_bnd0 l i: i \in l -> fib (i.+2) <= Zeck_val l.
Lemma Zeckv_bnd l i: i \in l -> i <= Zeck_val l.
Lemma Zeckvp_bnd l i: i \in l -> i <= Zeck_valp l.
```

```
Lemma Zeck_val_cv0 B (t:{set 'I_B.+1}) (f:nat->nat):
  \sum_(i <- (seto_to_seq t)) f i =  \sum_(i in t) f i.
Lemma Zeck_val_cv1 m (t:{set 'I_m.+1}) :
  Zeck_val (seto_to_seq t) =  Zeck_sval t.
Lemma Zeck_valp_cv1 B (t:{set 'I_B.+1}) :
   Zeck_valp (seto_to_seq t) =  Zeck_svalp t.

Lemma Zeck_val_cv2 B (n := Zeck_val l): uniq l -> n <= B ->
   n =  Zeck_sval (seq_to_seto m l).
Lemma Zeck_valp_cv2 l B (n := Zeck_valp l): uniq l -> n <= B ->
  n =  Zeck_svalp (seq_to_seto B l).
```

We consider now $\mathscr{R}_B(n)$ and $\mathscr{A}_B(m,n)$ the set of all sets E in $\mathfrak{P}(I_{B+1})$ such that $Z_E = n$, and in the second case, we moreover assume $Z'_E = m$. The cardinal of these sets will be denoted by $R_B(n)$ and $A_B(m,n)$. We shall define $\bar{R}(n) = R_n(n)$ and $\bar{A}(m,n) = A_n(m,n)$. Informally $\bar{R}(n)$ is the number of representations of n since each representation can be uniquely associated to set E of $\mathfrak{P}(I_{n+1})$.

```
Definition GRr B n := #|[set t:{set 'I_B.+1} | Zeck_sval t == n ]|.
Definition GAr B m n := #|[set t:{set 'I_B.+1} |
   (Zeck_svalp t == m) && (Zeck_sval t == n) ]|.

Definition GR n := GRr n n.
Definition GA m n := GAr n m n.
```

Assume $A \le B$, and let g be the function that associates to each element of $\mathfrak{P}(I_{A+1})$ an element of $\mathfrak{P}(I_{B+1})$ by converting each element. Then g is injective, and $Z_{g(E)} = Z_E$ (instead of Z, every other function could be used, for instance Z′). Thus g induces a bijection $\mathscr{R}_A(n)$ into $\mathscr{R}_B(n)$, and the sets have the same cardinal; it follows

$$R_A(n) = R_B(n) \quad \text{if} \quad n < F_{A+3},\ A \le B \tag{2.3}$$

i.e., if elements of any representation of n are $\le A$ and $\le B$. In particular if $n \le B$, then $R_B(n) = \bar{R}(n)$. The same holds for $A_B(m,n)$, provided that $m < F_{A+2}$ or $n < F_{A+3}$.

```
Lemma GARr_aux n B
  (g: ({set 'I_n.+1} ->  {set 'I_B.+1})
     := (fun t => [set (inord (i:'I_n.+1)) | i in  t])):
  n <= B ->
  (injective g /\ forall (t:{set 'I_n.+1}) (F: nat -> nat),
     \sum_(i in t) F i= \sum_(i in (g t)) F i).
Lemma GRr_big0 A B n: n < fib A.+3 -> A <= B -> GRr A n = GRr B n.
Lemma GAr_big0 A B m n:  ((m < fib A.+2) || (n < fib A.+3)) -> A <= B ->
    GAr A m n = GAr B m n.
Lemma GRr_big B n: n <= B -> GRr B n = GR n.
Lemma GAr_big B m n: n <= B -> GAr B m n = GA m n.
```

If $A_B(m,n)$ is non-zero, then $\mathscr{A}_B(m,n)$ is nonempty, hence has an element E such that $m = Z'_E$ and $n = Z_E$. This implies $m = e(n)$ (the result has been shown for a list, but applies to a set). Conversely, it is clear that $A_B(e(n),n) = R_B(n)$. It follows

$$\bar{A}(e(n),n) = \bar{R}(n). \tag{2.4}$$

Assume $F_{B+4} \leq n+1$; in this case n has no representation where all elements are $\leq B$, so that $R_B(n) = 0$. Assume $n = 0$ or $n = 1$; then $Z_l = n$ means that l is empty or the list [0]; it easily follows $R_B(0) = R_B(1) = 1$. One deduces that $R_0(n)$ is one when $n \leq 1$, zero otherwise.

We have $R_{B+1}(n) = R_B(n) + R_B(n - F_{B+3})$, where the second term is zero when $n < F_{B+3}$. We write this formally as

$$R_{B+1}(n) = R_B(n) + R_B(n - F_{B+3}) \cdot (n \geq F_{B+3}). \tag{2.5}$$

The case $n < F_{B+3}$ follows from (2.3). Let E be any set such that $Z_E = n$. Then $B+1$ may or may not be in E. Thus $\mathscr{R}_{B+1}(n)$ can be partitioned in to disjoint subsets X and Y. Note that, if g is as above, its image of $\mathscr{R}_B(n)$ is exactly X (the set of sets that do not contain $B+1$). If E is in Y, it contains $B+1$; removing it gives us a set E', not containing $B+1$ such that $Z_{E'} = Z_E - F_{B+3} = n - F_{B+3}$. If $g(E)$ is $g(E)$ with $B+1$ added, that g' is also injective, and it image of $\mathscr{R}_B(n - F_{B+3})$ is Y. Similarly we have

$$A_{B+1}(m,n) = A_B(m,n) + A_B(m - F_{B+2}, n - F_{B+3}) \cdot (n \geq F_{B+3})(m \geq F_{B+2}). \tag{2.6}$$

```
Lemma GAr_notz B m n: GAr B m n!= 0 -> m = Zeckp n.
Lemma GARr_e B n: GAr B (Zeckp n) n = GRr B n.
Lemma GAR_e n: GA (Zeckp n) n = GR n.
Lemma GRr_notz B n: fib B.+4 <= n.+1 -> GRr B n = 0.
Lemma GRr_b0 B:  GRr B 0 = 1.
Lemma GRr_b1 B:  GRr B 1 = 1.
Lemma GRr_0n n:  GRr 0 n = (n<=1).
Lemma GRr_Sbn B n :
  GRr B.+1 n =  GRr B n + (GRr B (n- fib(B.+3))) * (fib(B.+3) <= n).
Lemma GAr_Sbn B m n:
  GAr B.+1 m n =  GAr B m n +
    (GAr B (m - fib (B.+2)) (n- fib(B.+3)))
    * (fib(B.+3) <= n) * (fib(B.+2) <= m).
```

Application $\bar{R}(10) = 2$. The representations are $10 = F_6 + F_3 = F_5 + F_4 + F_2$.

```
Lemma GR_example: GR 10 = 2.
Proof.
have Ha n: GRr 1 n = (n <= 3).
  by rewrite GRr_Sbn !GRr_0n; case:n => [ | [| [ | [ | n]]]].
have ha: GRr 3 2 = 1 by rewrite -(@GRr_big0 1 3 2) // GRr_Sbn ! GRr_0n.
rewrite /GR; rewrite -(@GRr_big0 4 10 10) // GRr_Sbn muln1 ha GRr_Sbn muln1.
by rewrite (@GRr_notz 2 10) // GRr_Sbn ! Ha.
Qed.
```

2.4 The function R

We shall now assume that the function $\bar{A}$ satisfies some recurrence relation. We start with an informal argument that justifies this claim. Consider

$$\Phi(x,y) = \prod_{n=1}^{\infty} (1 + x^{F_n} y^{F_{n+1}}) = \sum_{m,n} A(m,n) x^m y^n. \tag{2.7}$$

The middle term is an infinite formal product, which will be named Φ; we can see it as the limit of the product of the k first terms. This is a polynomial, and we can write it uniquely as

$\sum_{m,n} A_k(m,n)x^m y^n$. For $k=0$ we have $A_k(0,0)=1$, all other coefficients being zero. We have the following recurrence

$$A_k(m,n) = A_{k-1}(m,n) + A_{k-1}(m-F_k, n-F_{k+1})$$

where it is understood as above that the second term is zero in case m or n are too small. In particular, if $k > \max(m,n)$ the quantity $A_k(m,n)$ becomes independent of k. The limit value will be denoted by $A(m,n)$. Since the recurrence relation is the same as (2.6) above, it follows $A_{k+1}(m,n)$ is equal to the quantity with the same name f the previous section, and that $A(m,n) = \bar{A}(m,n)$. Note that similarly $\Phi(1,y) = \sum R(n)y^n$ where $R = \bar{R}$.

Consider the magic formula

$$(1+xy)\Phi(x,xy) = \Phi(y,x).$$

Obviously, the LHS is the product of all $x^\alpha (xy)^\beta$, where the exponents are as above, starting with $n=0$. This is also the product of the $x^{\alpha+\beta}y^\beta$. Since $\alpha = F_n$ and $\beta = F_{n+1}$ we get $\alpha+\beta = F_{n+2}$, thus establishing the formula. Computing the coefficients of $x^m y^n$ yields

$$A(m,n) = A(n-m,m) + A(n-m,m-1) \tag{2.8}$$

where it is understood that both quantities are zero when $n < m$, the second is zero when $m < 1$. Moreover $A(0,0) = 1$. Its is rather easy to show that this equation has a unique solution. The idea now is to define a function A satisfying this recurrence relation, and a function R by $R(n) = A(e(n), n)$, by analogy with (2.4).

We use an auxiliary function with an additional argument to ensure termination, then show that the function A satisfies the specifications, and that R satisfies

$$R(n) = A(n-(en), e(n)) + A(n-e(n), e(n)-1). \tag{2.9}$$

```
Fixpoint FA_rec (m n k:nat) :=
   if k is k'.+1 then
    if (m==0) then (n==0): nat
    else if n < m then 0  else FA_rec (n-m) m k' + FA_rec (n - m) (m-1) k'
  else 0.
Definition FA m n := FA_rec m n (m+n).+1.
Definition FR n := FA (Zeckp n) n.

Lemma FA_00: FA 0 0 = 1.
Lemma FA_0S n:  FA 0 n.+1 = 0.
Lemma FA_small m n: n < m -> FA m n = 0.
Lemma FA_gen m n: FA m.+1 (m.+1+n) = FA n m.+1 + FA n m.
Lemma FA_mm m: FA m.+1 m.+1 = FA 0 m.
Lemma FA_mm' m: FA m m = ((m <=1):nat).

Lemma FR_0: FR 0 = 1.
Lemma FR_1: FR 1 = 1.
Lemma FR_rec n (e := Zeckp n): n != 0 -> FR n = FA(n -e) e + FA (n-e) e.-1.
```

Consider now the relations (2.1). By induction, using the last two equations, we get that $A(m,n)$ can be non-zero only if $m = e(n)$. In particular, it is zero when $m = e(n)+1$ or $m+1 = e(n)$. Assume $0 \in \mathrm{Zeck}(n)$. This means that $n = F_2 + F_{a+4} + \ldots$, so that $e(n) = F_2 + F_{a+3} + \ldots$ and $e(e(n)) = F_2 + F_{a+2} + \ldots$. We deduce, $e(n) = e(n-1)+1$ and $e(e(n)) = e(e(n-1))+1$. This gives $R(n) = R(e(n-1))$. If zero is not in $\mathrm{Zeck}(n)$, we get $n - e(n) = e(e(n))$; we deduce $R(n) = R(e(n)) + x$, where x is some expression studied below. By induction, it follows $R(n) > 0$. This can be restated as: $A(m,n)$ is non-zero if and only if $m = e(n)$.

```
Lemma FA_nz m n:  FA m n != 0 -> m = Zeckp n.
Lemma FA_prop0 a b: Zeckp b = a.+1 -> FA a b = 0.
Lemma FA_prop0bis a b: (Zeckp b).+1 = a -> FA a b = 0.

Lemma Zeckp_prop5 n (e := Zeckp): Zeck_li n.+1 = 0 ->
  (n == 0) ||
  [&& (Zeck_li n != 0), e (e n.+1) == (e (e n)).+1 & (e (n.+1) == (e n).+1)].
Lemma FR_prop0 n: Zeck_li n != 0 ->
  FR n = FR (Zeckp n) + FA (Zeckp (Zeckp n)) (Zeckp n).-1.
Lemma FR_prop1 n: Zeck_li n.+1 = 0 -> FR n.+1 = FR (Zeckp n).
Lemma FR_positive n: 0 < FR n.
```

2.5 Properties of R

We have

$$\mu(n) \text{ odd} \implies \mathrm{R}(n) = \mathrm{R}(e(n)). \tag{2.10}$$

Proof: the second relation of (2.1) says $\mathrm{R}(n) = \mathrm{R}(e(n)) + \mathrm{A}(e(y)+1, y)$ where $y = e(n)-1$. The second term cannot be zero, so that $\mathrm{R}(n) = \mathrm{R}(e(n))$. It follows

$$\mu(n) \text{ even non-zero} \implies \mathrm{R}(n) = \mathrm{R}(e(n)) + \mathrm{R}(e(n)-1) \text{ and } \mathrm{R}(e(n)) = \mathrm{R}(e^2(n)). \tag{2.11}$$

Here, and below, $e^q(n)$ means the q-th iteration of e to n. The second relation follows from (2.10) as $\mu(e(n)) = \mu(n)-1$. Since $\mu(n)$ is even, we can write $n = m + \mathrm{F}_{2k+2}$, so that $e(n) = e(m) + \mathrm{F}_{2k+1}$ and $e(e(n)) = e(e(m)) + \mathrm{F}_{2k}$. Since $\mu(n)$ is non-zero we have $\mathrm{R}(n) = \mathrm{R}(e(n)) + \mathrm{A}(e(e(n)), e(n)-1)$. We conclude by $e(e(n-1)) = e(m) + e(\mathrm{F}_{2k+1}-1) = e(e(n))$. One deduces

$$\mathrm{R}(\mathrm{F}_{2k+1}) = \mathrm{R}(\mathrm{F}_{2k}) = \min(1,k); \qquad \mathrm{R}(\mathrm{F}_i - 1) = 1.$$

This is one of the results of section 4 of [5], it comes as a special case of:

$$\mathrm{R}(n) = \mathrm{R}(e^{k+1}(m)) + (k/2)\cdot \mathrm{R}(e^k(m)), \qquad k = \mu(n) \qquad m = n - \mathrm{F}_{k+2}. \tag{2.12}$$

(This is the main result of section 3 of [5].) By (2.10) it suffices to consider the case where k is even. The equation holds for $n = 0$ by accident, so we may assume n non-zero. By (2.11), the LHS is $\mathrm{R}(e(n)) + \mathrm{R}(e(n)-1)$. The result follows by induction and

$$\mathrm{R}(e(n)-1) = \mathrm{R}(e^{2k+2}(m)) \text{ if } n = \mathrm{F}_{2k+2} + m, 2k+4 \le \mu(m).$$

Note $e(n) = \mathrm{F}_{2k+1} + e(m)$ and $e^2(n) = \mathrm{F}_{2k} + e^2(m)$. Since $\mathrm{F}_{2k+1} - 1$ is the sum of even-based Fibonacci numbers, we can write $e(n) - 1 = 1 + x$; so that $\mathrm{R}(1+x) = \mathrm{R}(e(x))$ with $e(x) = \mathrm{F}_3 + \ldots + \mathrm{F}_{2k-1} + e(e(m))$. Now $\mathrm{R}(e(x)) = \mathrm{R}(e(e(x)))$. So $\mathrm{R}(e(n)-1) = \mathrm{R}(\mathrm{F}_2 + \ldots + \mathrm{F}_{2k-2} + e^3(m))$. The result follows by induction on k.

```
Definition Zeckpn k n:= iter k Zeckp n.

Lemma FR_prop3 n: odd (Zeck_li n) -> FR n = FR (Zeckp n).
Lemma FR_prop4 m: llen 1 (Zeck_li m) -> FR (Zeckp m).+1 = FR (Zeckp (Zeckp m)).
Lemma FR_fib1 k: FR (fib k.*2.+3) = FR (fib k.*2.+2).
Lemma FR_fib2 k: FR (fib k.*2.+1) = FR (fib k.*2).
Lemma FR_fib3 k: FR (fib k).-1 = 1.
Lemma FR_split n (p := fib (Zeck_li n).+2) (m := n - p):
  n != 0 -> n = p + m /\  (m = 0 \/  llen (Zeck_li n) (Zeck_li m)).
```

```
Lemma Zeckp_prop6 n m q (s := pred_seq (Zeck m)):
  n = m + fib q.+3 -> (m = 0 \/  llen q.+1 (Zeck_li m)) ->
  [/\ Zeck (Zeckp n) = rcons (Zeck (Zeckp m)) q,
     (q >0 -> Zeck (Zeckp (Zeckp n)) = rcons (Zeck (Zeckp (Zeckp m))) q.-1),
     sorted llen (q :: rev s),  Zeckp m = Zeck_val (rev s) &
    Zeckp n = fib q.+2 +  Zeckp m].
Lemma FR_prop5 n: ~~ odd (Zeck_li n) -> (Zeck_li n) != 0 ->
  FR n =  FR (Zeckp n) + FR((Zeckp n).-1) /\ FR (Zeckp n) = FR (Zeckpn 2 n).
Lemma FR_fib4 k: FR (fib k.*2) = k.-1.+1.

Lemma FR_prop6 k m: llen (k.+1).*2 (Zeck_li m) ->
   FR (fib (k.+1).*2.+1 + Zeckp m).-1 = FR (Zeckpn (k.+1).*2 m).
Lemma FR_prop7 n (k :=Zeck_li n) (m := n -  fib (k.+2)) :
  ~~(odd k) -> FR n = FR (Zeckpn k.+1 m) + (k./2) * FR (Zeckpn k m).

Theorem FR_fib k: FR (fib k) = (k./2).-1.+1.
Theorem FR_prop8 n (k :=Zeck_li n) (m := n -  fib (k.+2)) :
  FR n = FR (Zeckpn k.+1 m) + (k./2) * FR (Zeckpn k m).
```

Let's show that $R(n) = 1$ if and only if n is the predecessor of a Fibonacci number (say $n = F_{k+2} - 1$ for some k). As noted above, this means that n has a unique representation. We have already shown one implication so let's proof the converse by induction on nk; the result holds for $n = 0$ with $k = 0$. Consider (2.12). It says $R(n) = R(e^{k+1}(m)) + (k/2) \cdot x$ where $x \geq 1$. Since the first term is > 0, we get $R(e^{k+1}(m)) = 1$ and $k/2 = 0$. By induction $e^{k+1}(m) = F_{q+2} - 1$ for some q. Depending on the parity of q, we get $m = \sum F_{2i+3+k}$ or $m = \sum F_{2i+4+k}$. The quantity n is the same, with F_{k+2} added. As the least index in the sum for m must be at least $k+4$ the first case is excluded (i.e., q must be even). So $n = \sum F_{2i+2+k}$, and $n = F_{(q+3+k)} - 1$.

```
Lemma FR_unique n: FR n = 1 -> exists k, n = (fib k.+2).-1.
```

We show here

$$R(F_{k+1} - 2) = k/2 \text{ provided } k \geq 2.$$

Even case: $n = F_{2k+4} - 2 = F_2 + F_5 + \ldots + F_{2k+3}$, so that $\mu(n) = 1$ and $R(n) = R(e(n-1)) = R(F_{2k+3} - 2)$. Odd case: We have $n = F_{2k+5} - 2 = F_4 + \ldots F_{2k+4}$, so that $\mu(n) = 2$. The main result says: $R(n) = R(e^3(m)) + R(e^2(m))$, where $m = n - F_4$. We have $e^2(m) = F_4 + \ldots F_{2k+2} = F_{2k+3} - 2$ and $e^3(m) = F_{2k+3} - 1$. Since $R(e^3(m)) = 1$, we get $R(F_{2k+5} - 2) = 1 + R(F_{2k+4} - 2)$. The result follows by induction.

```
Lemma Zeck_fib3 k:
   Zeck (fib (k.+2).*2).-2 = rev(0 :: mkseq (fun i=> i.*2.+3) k).
Lemma Zeck_fib4 k:
   Zeck (fib k.*2.+3).-2 = rev (mkseq (fun i=> i.*2.+2) k).
Lemma FR_fib5 k: FR (fib k.*2.+4).-2 = FR(fib k.*2.+3).-2.
Lemma FR_fib6 k: FR (fib (k.+2).*2.+1).-2 = (FR (fib (k.+2).*2).-2).+1.
Theorem FR_fib7 k:  FR (fib k.+3).-2 = (k.+2)./2.
```

We consider here some applications. We first consider the case where $n = F_a + F_b$, then the case $F_a + F_b + F_c$. This one is a bit more complicated. The formula simplifies when F_a is 1 or 2 (case $k/2 - 1 = 0$). A special case is $4F_i = F_{i_2} + F_i + F_{i+2}$. Here $(j - k - 1)/2 = 0$.

```
Lemma FR_fib_sum1 i k: 2 <= k ->
```

```
  FR(fib (i+ k.+2) + fib k) = (i.+3)./2 + (k./2).-1 * (i.+4)./2.
Lemma FR_fib_sum2 i j k : 2 <= k -> k.+2 <= j -> j.+2 <= i ->
  FR(fib i + fib j + fib k) = (i-j+1)./2 + ((j-k-1)./2) * (i-j+2)./2
    + (k./2.-1)*( (i-j+1)./2 + (j-k)./2 * (i-j+2)./2).
Lemma FR_fib_sum3 i j : 4 <= j -> j.+2 <= i ->
  FR(fib i + fib j + 1) = (i-j+1)./2 + (j-3)./2 * (i-j+2)./2.
Lemma FR_fib_sum4 i j : 5 <= j -> j.+2 <= i ->
  FR(fib i + fib j + 2) = (i-j+1)./2 + (j-4)./2 * (i-j+2)./2.

Lemma fib_times2 i: (fib i.+2) * 2 = fib i.+3 + fib i.
Lemma fib_times3 i: (fib i.+2) * 3 = fib i.+4 + fib i.
Lemma fib_times4 i: (fib i.+2) * 4 = fib i.+4 + fib i.+2 + fib i.

Lemma FR_fib_sum5 i: FR(fib (i + 4) + 1) = (i.+3)./2.
Lemma FR_fib_sum6 i: FR(fib (i + 5) + 2) = (i.+3)./2.
Lemma FR_fib_sum7 i: FR(fib (i + 6) + 3) = (i.+3)./2 + (i.+4)./2.
Lemma FR_fib_sum8 i: FR(fib (i + 6) + 4) = (i.+3)./2.
Lemma FR_fib_sum9 i: FR ((fib i.+4) * 2) = ((i./2)*2).+2.
Lemma FR_fib_sum10 i: FR ((fib i.+4) * 3) = ((i./2)*3).+2.
Lemma FR_fib_sum11 i: FR ((fib i.+4) * 4) = ((i./2)*3).+1.
```

As $L_k = F_{k-1} + F_{k+1}$ the expression for $R(L_k)$ is easy. Similarly for $L_{2j}F_k = F_{k+2j} + F_{k-2j}$.

```
Lemma FR_lucas1 n: FR (lucas n.+3) = n./2.*2.+1.
Lemma FR_lucas2 n: FR (lucas n.*2.+3) = n.*2.+1.
Lemma FR_lucas3 n: FR (lucas n.*2.+4) = n.*2.+1.
Lemma FR_lucas4 k j: j.*2.+2 <= k -> 1 <= j ->
 FR (lucas (j.*2) * (fib k)) = j.*2 + (j.*2.+1)*(k./2-j-1).
```

Chapter 3

The fusc function

The objective of this chapter is to study the properties of the sequence defined by the following property

$$s_0 = 1, \quad s_1 = 1, \quad s_{2n} = s_n, \quad s_{2n+1} = s_n + s_{n+1} \qquad (n > 0). \tag{3.1}$$

In his first paper, Dijkstra [6] introduces this function (for $n > 0$), and says that "this definition, as far as complexity is concerned, seems to lie between the Fibonacci series and the Pascal triangle". In fact, these three series are linked in a non-trivial way. In the third paragraph, he gives an iterative definition (equation (3.14)). The third paragraph is:

(The function fusc is of a mild interest on account of the following property: with $f_1 = \text{fusc}(n_1)$ and $f_2 = \text{fusc}(n_2)$ the following two statements hold: "if there exists an N such that $n_1 + n_2 = 2^N$, then f_1 and f_2 are relatively prime" and "if f_1 and f_2 are relatively prime, then there exists an n_1, an n_2, and an N such that $n_1 + n_2 = 2^N$". In the above recursive definition, this is no longer obvious, at least not to me; hence its name.)

In the second paper, he explains that reversing the order of bits leaves the fusc-value unchanged. Thus $s_{25} = s_{19}$ as $r_2(25) = 19$. In the same fashion $s_{24} = s_3$. He also explains that another function, denoted $i(n)$ below, leaves the fusc-value unchanged. Since $i(19) = 29$ we get $s_{19} = s_{29}$. As $29 + 3 = 2^5$, the two quantities s_{29} and s_3 have be coprime, according to the previous paragraph. This means that s_{25} and s_{24} are coprime. One can restate the previous paragraph as: "if $n_1 = n_2 + 1$, then f_1 and f_2 are relatively prime" and "if f_1 and f_2 are relatively prime, then there exists an n_1 and an n_2, such that $n_1 = n_2 + 1$". Did Dijkstra try to obfuscate this property? did he know that this amounts to prove that every positive rational number can be uniquely written in the form s_n/s_{n+1}?.

We shall sometimes consider the sequence as a table, denoted $\mathbf{T}$ here (this will be generalized later on under the name S). We have $\mathbf{T}_{ij} = \mathbf{T}(i, j) = \mathbf{T}_i(j) = s_n$ where $n = 2^i + j$, and the constraint is $j < 2^i$. It is sometimes useful to extend the table, allowing j to be equal to 2^i; then $\mathbf{T}(i, 2^i)$, the last element of row i, is $\mathbf{T}(i + 1, 0)$, the first element of the next row.

Note that s_0 does not appear in $\mathbf{T}$ (it does not participate in the recurrence, and did not appear in the first paper of Dijkstra). Each row of $\mathbf{T}$ is symmetric (from which one deduces $s_{19} = s_{29}$) and has $2^i + 1$ elements, the first and last elements are 1, the middle element is 2. An element at even position in $\mathbf{T}_{i+1}$ is the copy of an element of $\mathbf{T}_i$, while an element at odd position is the sum of its two neighbors.

3.1 Definition

This is the translation of (3.1):

```
Definition fusc_prop f:=
  [/\ f 0 = 0, f 1 = 1,
   (forall n, f (n.*2) =  f n) &
   (forall n, f (n.*2.+1) =  f n + f (n.+1)) ].
```

We first show an induction principle: in order for P to be true for all integers, it suffices that P holds for 0, 1, and that P(n) implies P($2n$), and that P(n) and P($n+1$) imply P($2n+1$). It follows trivially that (3.1) has at most one solution.

```
Lemma fusc_ind P:
   P 0 -> P 1 ->
   (forall n, P n -> P (n.*2)) ->
   (forall n, P n -> P n.+1 -> P (n.*2.+1)) ->
   forall n, P n.
Lemma fusc_unique f g: fusc_prop f -> fusc_prop g -> f =1 g.
```

We define the function by induction on a second hidden parameter.

```
Definition fusc n :=
  let fix loop  n k :=
    if k is k'.+1 then
      if n<=1 then n else if (odd n) then (loop n./2 k') + loop (uphalf n) k'
      else loop n./2 k'
    else 0
  in loop n n.
```

This function satisfies equation (3.1). Note also that $s_{2n-1} = s_{n-1} + s_n$ (even if $n = 0$).

```
Lemma fusc_rec: fusc_prop fusc.
Lemma fusc0: fusc 0 = 0.
Lemma fusc1: fusc 1 = 1.
Lemma fusc2: fusc 2 = 1.
Lemma fusc3: fusc 3 = 2.
Lemma fusc_even n: fusc n.*2 = fusc n.
Lemma fusc_odd n: fusc n.*2.+1 = fusc n + fusc n.+1.
Lemma fusc_podd n: fusc n.*2.-1 = fusc n.-1 + fusc n.
```

By fusc-induction, it is clear that s_n is zero only when $n = 0$, and s_{n+2} can be obtained from s_n and s_{n+1} via the following relation:

$$s_{n+2} = (s_n + s_{n+1}) - 2(s_n \bmod s_{n+1}). \tag{3.2}$$

One deduces:

$$s_n = f(s_{n+1}, s_{n+2}) \qquad \text{where } f(x,y) = \begin{cases} y - x & \text{when } x \text{ divides } y \\ x + y - 2(y \bmod x) & \text{otherwise.} \end{cases} \tag{3.3}$$

(the definition below is a bit different). One deduces injectivity of $i \mapsto (s_i, s_{i+1})$.

```
Definition fusc_prev x y := x * ((y %/x).+1 - (y%%x==0).*2) - y%% x.

Lemma fusc_eq0 n: (fusc n == 0) = (n == 0).
Lemma fusc_gt0 n: (0 < fusc n) = (0 < n).
Lemma fusc_monotone n: odd n -> n = 1 \/ fusc n.+1 < fusc n.
Lemma fusc_normal_rec n:
   fusc (n.+2) + ((fusc n) %% fusc (n.+1)).*2 = (fusc n) + fusc (n.+1).
Lemma fusc_prevE n (P:= fun x y =>  x * ((y %/x).+1 - (y%%x==0).*2) - y%% x):
 fusc n = P (fusc n.+1)(fusc n.+2).
Lemma fuscp_injective n m:  fusc n = fusc m -> fusc n.+1 = fusc m.+1 -> n = m.
```

We have $\mathbf{T}(i,0) = 1$, so that $\mathbf{T}(i,2^i) = 1$. We have $\mathbf{T}(i,1) = i+1$ and $\mathbf{T}(i,-1) = i-1$. Each row is symmetric (a palindrome), in the sense that if $j+k = 2^i$, then $\mathbf{T}_i(j) = \mathbf{T}_i(k)$. In terms of the fusc function, this is

$$p = 2^n, \quad p \le a,b \le 2p, \quad a+b = 3p \implies s_a = s_b. \tag{3.4}$$

For every j, the sequence formed of the elements of column j of the Stern table $\mathbf{T}$ forms an arithmetic progression whose common difference is s_j. This is equivalent to

$$s_{2^{i+1}+j} = s_{2^i+j} + s_j. \tag{3.5}$$

We deduce the strange relation: $s_{p-i} + s_i = s_{p+i}$ valid when $i \le p = 2^n$. It implies $s_i \le s_{p+i}$ (with equality only when $i = p$). The last result stated here is that every element at odd location of row i to $\mathbf{T}$ is at least $i+1$. From $\mathbf{T}(i,1) = i+1$, this also: $\min_j \mathbf{T}(i,2j+1) = i+1$.

```
Lemma fusc_pow2 n: fusc (2^n) = 1.
Lemma fusc_succ_pow2 n: fusc ((2^n).+1) = n.+1.
Lemma fusc_pred_pow2 n: fusc ((2^n).-1) = n.
Lemma fusc_palindrome i a b (p := 2 ^ i):
   p <= a <= p.*2 -> a + b = 3 * p -> fusc a = fusc b.
Lemma fusc_col_progression i j: j <= 2 ^ i ->
  fusc(2^ (i.+1) + j) = fusc(2 ^ i + j) + fusc j.
Lemma fusc_symmetry1 i n: i <= 2^n -> fusc(2^n -i) + fusc i = fusc (2^n + i).
Lemma fusc_bound i n: i <= 2 ^n ->
  fusc i <= fusc (2 ^ n + i) ?= iff (i == 2 ^ n).
Lemma fusc_lower_bound  n:  n.*2 < 2 ^ fusc n.*2.+1.
```

By fusc-induction it obvious that s_{n+1} is coprime to s_n. The palindrome condition then says: if $a \ge p$, $b \ge p$, $a+b+1 = 3p$, then s_a and s_b are coprime. We show again that $n \mapsto (s_n, s_{n+1})$ is injective: assume $s_i = s_j$ and $s_{i+1} = s_{j+1}$. Since $s_i < s_{i+1}$ equivalent to i even, the two numbers i and j have the same parity. Assume i even, the other case being similar; say $i = 2n$, $j = 2m$. The first relation gives $s_n = s_m$; the second gives $s_n + s_{n+1} = s_m + s_{m+1}$ so that $s_{n+1} = s_{m+1}$; the result is then obvious by induction.

```
Lemma fusc_coprime n: coprime (fusc n) (fusc n.+1).
Lemma fusc_coprime_palindrome  i a b (p := 2 ^ i):
   p <= a <= p.*2 -> (a + b).+1 = 3 * p -> coprime (fusc a)(fusc b).
Lemma fusc2_injective n m:  fusc n = fusc m -> fusc n.+1 = fusc m.+1 -> n = m.
```

Since s_n and s_{n+1} are coprime, there is a Bezout relation. Of course, it is not unique. One solution is

$$s_{p+n}s_{n+1} = 1 + s_{p+n+1}s_n \qquad (n < p, p = 2^k) \tag{3.6}$$

We shall discuss it later. More generally consider

$$a_n s_{n+1} = 1 + c_{n+1} s_n.$$

It is rather easy to find a sufficient induction property, for instance

$$a_{2n} = a_n, \quad c_{2n} = c_n, \quad a_{2n+1} = a_n + c_{n+1}, \quad c_{2n+3} = a_{n+1} + c_{n+2}.$$

We have $a_0 = 1$ but the value of c_1 is arbitrary. Consider the case $c_1 = 0$. It is not clear why the equation would have an explicit solution. It is easy to see that c_n is 0 when n is a power of two, and is a_n otherwise (in the alternate formulation (3.6) the equivalent of c_n is always equal to the equivalent of a_n and is never zero). Define $\mathrm{T}_{ij} = a_n$ where $n = 2^i + j$, and the constraint is $j < 2^i$. We have $\mathrm{T}_{i+1,2j} = \mathrm{T}_{i,2j}$, $\mathrm{T}_{i+1,2j+1} = \mathrm{T}_{i,j} + \mathrm{T}_{i,j+1}$, so that T satisfies the same properties as **T**, with one exception: $\mathbf{T}(i,2^i) = 1$ while $\mathrm{T}(i,2^i) = 0$. We shall study the Stern table in the next chapter, and prove that $\mathbf{T} = \mathrm{S}(1,1)$ while $\mathrm{T} = \mathrm{S}(1,0)$. There is an explicit formula for the elements of $\mathrm{S}(m,n)$, it gives $a_n = s_{2^{\log_2 n}-n}$. Its is straightforward to show that this a_n (and the corresponding c_n) satisfies the recurrence relation, thus the Bezout relation.

```
Lemma fusc_bezout
   (a := fun  n =>fusc (2^log2 n - n))
   (c := fun n => if (n== 2^(log2 n).-1) then 0 else a n):
   (forall n,  (c n.+1) * (fusc n) + 1 = (a n) *(fusc n.+1)).
Lemma fusc_bezout2 i n: i < 2^n ->
  fusc (2 ^ n + i) * fusc i.+1 = 1 + fusc (2 ^ n + i.+1) * fusc i.
```

3.2 Bijection between N and Q

In [3] (first paper, section 9), Cantor proves that an ordered set is order isomorphic to the interval $]0,1[$ of **Q** (thus to $]0,\infty[$ or **Q**) if the set is infinite countable, totally ordered, has no least element, no greatest element, and for any x and y such that $x < y$, there is z such that $x < z < y$. A set E is infinite countable if there exists a bijection $\mathbf{N} \to \mathrm{E}$. By the Cantor Bernstein theorem (conjectured by Cantor), E is infinite countable if there is an injection $\mathrm{E} \to \mathbf{N}$ and an injection $\mathbf{N} \to \mathrm{E}$. This is true for **Q**, as $a/b \mapsto (a,b)$ is injective (if a/b is a reduced fraction), and $(n,m) \mapsto n + \binom{n+m+1}{2}$ is a bijection $\mathbf{N}\times\mathbf{N} \to \mathbf{N}$. On the other hand, $]0,1[$ is order isomorphic to **N** (if $x = a/b$ and $y = c/d$, say $x <_q y$ when the pair $(a+b,a)$ is lexicographically less than the pair $(c+d,c)$; this gives a well-ordered set whose ordinal is ω_0). If one wants a bijection $\mathrm{E} \to \mathbf{N}$ (where E is $]0,1[$, but the cases $]0,\infty[$ or **Q** are similar), one can proceed as follows: let $x = a/b$, and count the number of $y = c/d$ such that $y <_q x$. By definition, this is the number of fractions with $c + d < a + b$, plus the number of fractions with $c + d = a + b$ and $c < a$ (if we relax the condition that a and b have to be coprime, and $0 < a < b$, the first number is the binomial coefficient that appears above, and the second is a).

The function $n \mapsto s_n/s_{n+1}$ is a bijection $\mathbf{N} \to \mathbf{Q}_+$ (the proof that is surjective is by induction on the well-ordering of **Q** given above). We give in this section an explicit inverse (here we prove by induction on **Q** that it is well-defined and injective).

Let's start with some small functions defined on **Q** (note: G_p and G_i are called N and D in [1, Exercise 6.24])

$$\mathrm{G}_p(x) = 1 + x, \qquad \mathrm{G}_i(1/(1+1/x)), \qquad \mathrm{G}_m(x) = x - 1, \qquad \mathrm{G}_j(x) = 1/(1/x - 1). \tag{3.7}$$

We have $G_i(x) = x/(1+x)$ and $G_j(x) = x/(1-x)$, assuming x non-zero. Obviously, G_m is the inverse of G_p. It happens that G_j is the inverse of G_i (note that $G_i(0) = 1$; so that there is a discontinuity here). We also have $1/G_j(x) = G_m(1/x)$.

```
Definition Sba_p (x: rat) := 1 + x.
Definition Sba_i (x: rat) := (1 + x^-1)^-1.
Definition Sba_m (x: rat) := x - 1.
Definition Sba_j (x: rat) := (x^-1 - 1)^-1.

Lemma Sba_i_prop x: x != 0 -> Sba_i x = x /(1 + x).
Lemma Sba_j_prop x: x != 0 -> Sba_j x = x /(1 - x).

Lemma Sba_pK: cancel Sbp_a Sbp_c.
Lemma Sba_mK: cancel Sbm_c Sba_p.
Lemma Sba_iK: cancel Sba_i Sba_j.
Lemma Sba_jK: cancel Sba_j Sba_i.
Lemma Sba_mjK x: (Sba_j x) ^-1 = (Sba_m x^-1).
```

If $x > 1$ then $G_m(x) > 0$ and if $0 < x < 1$ then $G_j(x) > 0$; conversely, every positive number y has the form $G_m(x)$ for $x > 1$ and the form $G_j(x)$ for $0 < x < 1$.

```
Lemma Sba_p_pos x: 0 < x -> 1 < (Sba_p x).
Lemma Sba_m_pos x: 1 < x -> 0 < (Sba_m x).
Lemma Sba_m_pos_contra x: 0 < x -> {y:rat | (1 < y) && (Sba_m y == x) }.
Lemma Sba_j_pos x: 0 < x < 1 -> 0 < (Sba_j x).
Lemma Sba_i_pos x: 0 < x -> 0 < Sba_i x < 1.
Lemma Sba_j_pos_contra x: 0 < x -> {y:rat | (0 < y <1) && (Sba_j y == x) }.
```

If $x = a/b$ is a rational number, we define $k(x)$ to be $a+b-1$. If $x \geq 0$ then $a \geq 0$ and $b \geq 1$ so that $k(x) \geq 0$. Note the use the absolute value to convert $a+b \in \mathbf{Z}$ into a natural number. If $x > 0$ then $a \geq 1$ so that $k(x) > 0$. If $x > 0$, then $k(x) = k(1/x)$. We deduce that $k(G_j(x))$ and $k(G_m(x))$ are $< k(x)$ under the conditions stated above.

```
Definition snumden' x := numq x + denq x.
Definition snumden x := `|snumden' x|.-1.
Lemma snumden_pos' x:  0 <= x -> 0 < snumden' x.
Lemma snumden_pos x: 0 <=x -> snumden' x = (snumden x).+1.
Lemma snumden_gt0 x: 0 < x ->  (snumden x) != 0%N.
Lemma snumden_inv x:  0 < x  -> snumden x = snumden (1/x).
Lemma Sba_m_snum x: 1 < x ->  (snumden (Sba_m x) <  snumden x)%N.
Lemma Sba_j_snum x: 0 < x <1 ->  (snumden (Sba_j x) <  snumden x)%N.
```

Let's consider the following iterative procedure. The procedure replaces x by $G_j(x)$ if $0 < x < 1$ or by $G_m(x)$ if $1 < x$, it stops otherwise. The return value of the function is a representation (for instance a list of booleans) of the choices made at every step. Note that $k(x)$ is strictly decreasing so that the procedure always stops.

We can convert this into a recursive function G' returning a `positive` value or into a function G returning a natural number. We use an auxiliary function that takes a second argument n and recurse on n; the result is independent of n if $k(x) < n$.

```
Fixpoint Stern_bij_rec (x:rat) (n:nat) :=
  if n is m.+1 then
    if(0 < x < 1) then xO (Stern_bij_rec (Sba_j x) m)
```

```
    else if(1 < x) then xI (Stern_bij_rec (Sba_m x) m)
    else xH
  else xH.
Definition Stern_bij1 x :=  Stern_bij_rec x (snumden x).+1.
Definition Stern_bij x := if (x<=0) then 0%N else nat_of_pos(Stern_bij1 x).

Lemma Stern_bij_recE (x:rat) n : 0 < x -> ((snumden x) < n)%N ->
  Stern_bij_rec x n  = Stern_bij1 x.
```

The following properties of G′ are trivial.

```
Lemma Stern_bij1_1: Stern_bij1 1 = 1%positive.
Lemma Stern_bij1_lt1 x : 0 < x < 1 ->
  Stern_bij1 x = xO (Stern_bij1 (Sba_j x)).
Lemma Stern_bij1_gt1 x : 1 < x ->
  Stern_bij1 x = xI (Stern_bij1 (Sba_m x)).
Lemma Stern_bij1_lt1K x : 0 < x ->
  Stern_bij1 (Sba_i x) = xO (Stern_bij1 x).
Lemma Stern_bij1_gt1K x : 0 < x ->
  Stern_bij1 (Sba_p x) = xI (Stern_bij1 x).
```

The properties of G are similar.

```
Lemma Stern_bij_0: Stern_bij 0 = 0%N.
Lemma Stern_bij_1: Stern_bij 1 = 1%N.
Lemma Stern_bij_lt1 x : 0 < x < 1 ->
   Stern_bij x =  (Stern_bij (Sba_j x)).*2.
Lemma Stern_bij_gt1 x : 1 < x ->
   Stern_bij x = (Stern_bij (Sba_m x)).*2.+1.
Lemma Stern_bij_lt1K x : 0 < x ->
  Stern_bij (Sba_i x) = (Stern_bij x).*2.
Lemma Stern_bij_gt1K x : 0 < x ->
  Stern_bij (Sba_p x) = (Stern_bij x).*2.+1.
```

The function G is the desired bijection. Clearly, G is surjective; moreover $0 < x$ implies $0 < \mathrm{G}(x)$. Thus G′ injective implies that G is injective. Assume $\mathrm{G}'(x) = \mathrm{G}'(y)$, where $x > 0$ and $y > 0$. We prove, by induction on an upper bound n of $k(x)$ and $k(y)$ that $x = y$. We have to consider the case where $x = 1$, $x > 1$ and $0 < x < 1$, similarly for y. Assume for instance $1 < x$. Then $\mathrm{G}'(x) = x_{\mathrm{I}}(\mathrm{G}'(\mathrm{G}_m(x)))$. Thus $\mathrm{G}'(x) = \mathrm{G}'(y)$ implies that y has to be > 1, so that $\mathrm{G}'(\mathrm{G}_m(x)) = \mathrm{G}'(\mathrm{G}_m(y))$. By induction it follows $\mathrm{G}_m(x) = \mathrm{G}_m(y)$, thus $x = y$.

```
Lemma Stern_bij1_surj p: { x | (0 < x) /\ (Stern_bij1 x = p) }.
Lemma Stern_bij_surj n: { x | (0 <= x) /\ (Stern_bij x = n) }.
Lemma Stern_bij_gt0 x: 0 < x -> (0 < Stern_bij x)%N.
Lemma Stern_bij1_inj  x y: 0 < x -> 0 < y  ->
  Stern_bij1 x = Stern_bij1 y -> x = y.
Lemma Stern_bij_inj  x y: 0 <= x -> 0 <= y  ->
  Stern_bij x = Stern_bij y -> x = y.
```

We show here that the inverse of G is the function F defined by $\mathrm{F}_n = s_n / s_{n+1}$. If $n = 2^p$, then $\mathrm{F}_n = 1/(n+1)$ and $\mathrm{F}_{n-1} = n$. In particular $\mathrm{F}_0 = 0$ and $\mathrm{F}_1 = 1$. However $\mathrm{F}_n > 0$ when $n > 0$. Moreover the recurrence relation (3.1) says

$$\mathrm{F}_{2n} = \mathrm{G}_i(\mathrm{F}_n) \quad \mathrm{F}_{2n+1} = \mathrm{G}_p(\mathrm{F}_n). \tag{3.8}$$

The palindrome condition can be restated as $1/\mathrm{F}_a = \mathrm{F}_b$ where $b = 3 \cdot 2^{(\log_2 a)-1} - a - 1$.

```
Definition Stern n := (fusc n)%:Q  / (fusc n.+1)%:Q.

Lemma Stern0: Stern 0 = 0.
Lemma Stern1: Stern 1 = 1.
Lemma Stern2: Stern 2 = 1/2%:Q.

Lemma Stern_ge0 n: 0 <= Stern n.
Lemma Stern_gt0 n: 0 < Stern n.+1
Lemma Stern_nz n: Stern n.+1 != 0.
Lemma Stern_pow2 n: Stern (2^n) = 1/(n.+1)%:Q.
Lemma Stern_pred_pow2 n: Stern ((2^n).-1) = n%:Q.
Lemma Stern_even n: Stern ((n.+1).*2) = Sba_i (Stern (n.+1)).
Lemma Stern_odd n: Stern (n.*2.+1) = Sba_p (Stern n).
Lemma Stern_palindrome a: (0 < a)%N ->
  (Stern a)^-1 = Stern ((3* 2^(log2 a).-1 -a).-1).
```

The function $n \mapsto \mathrm{F}_n$ is injective: since s_n and s_{n+1} are coprime, it suffices to show that the two relations $s_n = s_m$ and $s_{n+1} = s_{m+1}$ imply $n = m$; this was proved above. It is surjective (proof by induction on $k(x)$). We deduce: if a and b are integers, coprime, and b is non-zero, there exists n such that $a = s_n$ and $b = s_{n+1}$.

There are simpler proofs: it is obvious, by induction on n, using (3.8), that $\mathrm{G}(\mathrm{F}_n) = n$. Replace n by $\mathrm{G}(x)$ and use injectivity of G. It follows: if $0 \le x$ then $\mathrm{F}_{\mathrm{G}(x)} = x$.

```
Lemma Stern_numden n:
  (numq(Stern n) == fusc n)  && (denq (Stern n) == fusc n.+1).
Lemma Stern_injective: injective Stern.
Lemma Stern_surjective x: 0 <= x -> {n | x = Stern n}.
Lemma fusc_pair_surj a b: coprime a b.+1 ->
  {n | a = fusc n /\ b.+1 = fusc n.+1 }.

Lemma SternK: cancel Stern Stern_bij.
Lemma Stern_bijK x: 0 <= x -> x = Stern (Stern_bij x).
```

Let y be the rational successor of x. This is the number that comes just after x in the enumeration of the rationals defined by G, thus is $\mathrm{G}^{-1}(\mathrm{G}(x)+1)$. Is there an explicit formula? Assume $x = \mathrm{F}_n$, so that $y = \mathrm{F}_{n+1}$. The question becomes: can we compute F_{n+1} from F_n. Since $\mathrm{F}_{n+1} = s_{n+1}/s_{n+2}$ and $\mathrm{F}_n = s_n/s_{n+1}$, the question becomes: can we compute s_{n+2} from s_n and s_{n+1}; the answer is given by equation (3.2). Thus, x and y are related by the function satisfying (1.6).

```
Lemma Stern_next n: Stern_succ_fct (Stern n) = Stern (n .+1).
Lemma Stern_prev n: Stern_prev_fct (Stern n.+1) = Stern n.
```

3.3 Other properties

Stern notes the following: assume that a, b and c are three consecutive numbers in the table (indexed by $n-1$, n and $n+1$ for instance). Then b divides the sum of a and c. This is obvious if n is odd as b is equal to the sum; more generally, the quotient is odd, say $2k+1$, and b appears k rows above (more precisely $b = s_{n/2^k}$). We have in fact

$$s_{n-1} + s_{n+1} = (2k+1)\,s_n \qquad \text{when } n = 2^k(2q+1). \tag{3.9}$$

This implies that $s_{n+1} = s_n + s_{n+2} \mod 2$. Thus, given three consecutive fusc numbers, exactly one of them is even (if two of them are even, so is the other one, and by induction all are even, contradicting $s_1 = 1$), and $s_0 = 0$ says that s_{3n} is even. We prove this property directly. Assume $a + c = b$; then a and c are coprime (note that $k = 0$).

```
Lemma fusc_middle_div k q  (n := 2^k * (q.*2.+1)):
    fusc (n.-1) + (fusc n.+1) = (k.*2.+1) * fusc n.
Lemma fusc_is_even1 n: fusc n + fusc n.+2 = fusc (n.+1) %[mod 2].
Lemma fusc_is_even n: (odd (fusc n)) = (n %%3 !=0).
Lemma fusc_coprime1 n: fusc n + fusc n.+2 = fusc (n.+1) ->
  coprime (fusc n) (fusc n.+2).
```

3.4 Fibonacci and fusc

We study in this section the maximum values of the row $\mathbf{T}_n$ of the Stern table. We pretend that the maximum is reached exactly twice, at positions near $l/3$ and $2l/3$, where l is the size of $\mathbf{T}_n$. If $n = 0$, the two elements of the row are 1 and 1, this is the only case where the maximum is one. If $n = 1$, the three elements of the row are 1, 2, and 1, this is the only case where the maximum is reached once. Let's consider

$$2^n \le c_{n+2} = \frac{4\cdot 2^n - (-1)^n}{3} < c'_{n+2} = \frac{5\cdot 2^n + (-1)^n}{3} \le 2.2^n. \tag{3.10}$$

We shall show that, for some c_n and c'_n, these relations are satisfied (as noted above, for $n = 1$, $<$ should be replaced by $=$).

We first define, by induction, a pair (c_n, d_n) satisfying the following relation:

$$c_{n+1} = 2d_n + 1, \qquad d_{n+1} = c_n + d_n, \qquad c_0 = d_0 = 0.$$

Eliminating one of the two variables gives:

$$c_{n+2} = 2c_n + c_{n+1}, \qquad d_{n+2} = 2d_n + d_{n+1} + 1.$$

Note that, since the solutions of $X^2 = 2 + X$ are -1 and 2, the following equation

$$X_{n+2} = 2X_n + X_{n+1}, \qquad X_0 = a, \quad X_1 = b, \tag{3.11}$$

has a unique solution of the form $X_n = \alpha 2^n + \beta(-1)^n$. In the case of c_n, the constants are $\alpha = -\beta = 1/3$. The equation for d_n is not linear but affine; hover $d_n + 1/2$ satisfies (3.11). We shall define $e_n = 3\cdot 2^{n-2} - c_n$; this satisfies also (3.11). We shall only consider e_n for $n \ge 2$ and give arbitrary integer values for $n < 2$.

```
Fixpoint Fib_fusc_rec n :=
   if n is m.+1 then let (a,b):= Fib_fusc_rec m in (b.*2.+1, a + b) else (0,0).
Definition Fib_fusc_sym n := 3*2^(n.-2) - (Fib_fusc_rec n).1.

Notation FMc n :=  (Fib_fusc_rec n).1.
Notation FMd n :=  (Fib_fusc_rec n).2.
Notation FMe n :=  (Fib_fusc_sym n).

Lemma FM_cE n: FMc (n.+1) =  (FMd n).*2.+1.
Lemma FM_dE n: FMd (n.+1) = (FMc n) + (FMd n).
Lemma FM_recc:
 [/\ FMc 0 = 0, FMc 1 = 1 & forall n, FMc (n.+2) = FMc (n.+1) + (FMc n).*2 ].
Lemma FM_recd:
 [/\ FMd 0 = 0, FMd 1 = 0 & forall n, FMd (n.+2) = FMd (n.+1) + (FMd n).*2.+1].
```

It happens that $c_n = d_n$ when n is even, $c_n = d_n + 1$ when n is odd. From this, we deduce a recurrence relation of order one (that depends on the parity of n), so that d_{n+1} has the same parity as n, and c_{n+1} is odd.

Note that the binary expansion of d_n is formed alternatively of ones and zeroes and it has exactly $n-1$ bits (for instance $d_5 = 10 = (1010)_2$ and $d_6 = 21 = (10101)_2$).

```
Definition zero_one_rep n:=
  nat_of_list(iter n (fun l => ~~(head true l) :: l) [::true]).

Lemma FM_recE1 n: FMc n = (FMd n)  + (odd n).
Lemma FM_recd2 n: FMd (n.+1) =  (FMd n).*2 + (odd n).
Lemma FM_recc2 n: FMc (n.+1) =  (FMc n - (odd n)).*2.+1.
Lemma FM_dodd n: odd (FMd (n.+1)) = odd n.
Lemma FM_codd n: odd (FMc (n.+1)).
Lemma FM_d_val n: FMd n.+2 = zero_one_rep n.
```

Note that $2^n \le d_{n+2} < 2^{n+1}$; thus $2^n \le c_{n+2} \le 2^{n+1}$. The same relation holds for e_n as $c_{n+2} + e_{n+2} = 3 \cdot 2^n$. We deduce from this a one-step recurrence relation for e_n, that clearly shows that e_n is odd. Thus, by induction, $c_n < e_n$ (there is equality for $n = 3$; in fact $c_n < 3 \cdot 2^{n-3} < e_n$). Moreover e_n satisfies equation (3.11).

```
Lemma FM_ecval n: FMc n.+2 + FMe n.+2 = 3 * 2 ^ n.
Lemma FM_cdbound n :  (FMd n) <=  FMc n <= (FMd n) .+1.
Lemma FM_dbound n :  2 ^ n <= FMd (n.+2) < 2^(n.+1).
Lemma FM_cbound n: 2 ^ n <= FMc (n.+2) <= 2^(n.+1).
Lemma FM_ebound n: 2 ^ n <= FMe (n.+2) <= 2^(n.+1).
Lemma FM_rece2 n: FMe n.+3 =  (FMe n.+2 + (odd n.+2)).*2.-1.
Lemma FM_rece n:  FMe (n.+4) = FMe (n.+3) + (FMe n.+2).*2.
Lemma FM_ltce n: (FMc (n.+4)).+1 < FMe (n.+4).
```

Let's show the two equalities in equation (3.10).

```
Lemma FM_c_value n: FMc n = if odd n then  ((2^n).+1) %/3 else ((2^n).-1)%/3.
Lemma FM_e_value n (a := 5*(2^n)):
   FMe n.+2 = if odd n then  (a.-1) %/3 else (a.+1)%/3.
```

We have $s(c_n) = \mathrm{F}_n$, easy by induction. One deduces $s(e_{n+2}) = \mathrm{F}_{n+2}$. We pretend that, if $k \le 2^{n+1}$, then $s_k \le \mathrm{F}_n$, where equality is strict, unless k is c_{n+2} or e_{n+2}. First, note that if k is even $s_k = s_{k/2}$, so that $s_k \le \mathrm{F}_{n-1}$. Assume $k = 2q+1$, so that $s_k = s_q + s_{q+1}$. One of q or $q+1$ is even, so that the sum is at most $\mathrm{F}_{n-1} + \mathrm{F}_{n-2} = \mathrm{F}_n$. In case of equality, we get that $q = c_{n-1}$, and $(q+1)/2 = c_{n-2}$ (or the same relation with q and $q+1$ exchanged, or the same relation with one or both c replaced by e). For small values of n, the result follows by case analysis; otherwise we can exclude the case where there is one c_i and one e_i, by using the relation $c_{n+4} + 1 \le e_{n+4} - 1$, together with $c_{n+1} = 2c_n \pm 1$, $e_{n+1} = 2e_n \pm 1$. The easy case is when there are two c_i or two e_i, which follows from (3.11).

```
Lemma FM_fuscc_val n: fusc (FMc n) = fib n.
Lemma FM_fusce_val n: fusc (FMe n.+2) = fib n.+2.

Lemma fib_bound a b n: a <=  fib n.+2  -> b <=  fib n.+1 ->
   a + b = fib n.+3  -> (a = fib n.+2) /\ (b = fib n.+1).
Lemma fusc_bound1 n k: k <= 2 ^ n -> fusc k <= fib (n.+1).
Lemma fusc_bound2 n k: k <= 2 ^ (n.+1) ->
  (fusc k <= fib n.+2 ?= iff ((k == (FMe n.+2))  || (k == (FMc n.+2)))).
```

3.5 Diagonals of the Pascal triangle

Let $b(n,k) = \binom{n}{k}$ be the solution of

$$\binom{n}{0} = 1, \quad \binom{0}{p+1} = 0, \qquad \binom{n+1}{p+1} = \binom{n}{p+1} + \binom{n}{p}. \tag{3.12}$$

The diagonal of the Pascal triangle is the set of all $b(n,k)$ where the sum $n+k$ is fixed. We have (lemma `bin_sum_diag` above):

$$\sum_{i<n} \binom{n-1-i}{i} = \mathrm{F}_n,$$

and we pretend that, if we reduce each term modulo two, then the sum becomes s_n. We start with

$$\binom{n+2}{k+2} = \binom{n}{k} + \binom{n}{k+2} \pmod 2.$$

This implies

$$\binom{2n}{2k+1} = 0, \quad \binom{2n}{2k} = \binom{2n+1}{2k+1} = \binom{2n+1}{2k} = \binom{n}{k} \pmod 2. \tag{3.13}$$

The result follows: let S_n be the sum. There are two cases to consider: n even and n odd. In each case, we use equation (1.1). The previous formulas say that the sums are $\mathrm{S}_n/2$ or $\mathrm{S}_{n/2+1}$ or zero.

```
Lemma bin_mod2_rec1 n k :
  'C(n.+2, k.+2) = 'C(n, k.+2) + 'C(n, k)  %[mod 2].
Lemma bin_mod2_rec2 n k: 'C(n.*2,k.*2) = 'C(n, k)  %[mod 2].
Lemma bin_mod2_prop1 n k: 'C(n.*2,k.*2.+1) %%2 = 0.
Lemma bin_mod2_prop2 n k: 'C(n.*2.+1,k.*2) = 'C(n, k)  %[mod 2].
Lemma bin_mod2_prop3 n k: 'C(n.*2.+1,k.*2.+1) = 'C(n, k)  %[mod 2].

Lemma bin_mod2_sum_diag (f := fun n i => 'C(n.-1-i, i) %% 2) n:
  \sum_(i<n) (f n i) = fusc n.
```

Let q be a prime number, $q_i(n)$ and $r_i(n)$ the quotient and remainder in the division of n by q^i. The exponent of q in the decomposition into prime numbers of $n!$ is $\sum q_i$, so that the exponent of q in $b(n,k)$ is $\sum_i (q_i(n) - q_i(k) - q_i(n-k))$. Each term here is ≥ 0, so that q divides $b(n,k)$ if and only if at least one term is non-zero. One can restate this as: there is i such that $r_i(n) < r_i(k)$. We show this result directly, using the relations $r_{i+1}(2a) = 2r_i(a)$, $r_{i+1}(2a+1) = 2r_i(a)+1$, established in the first chapter.

```
Lemma binom_evenP n k:  ~~(odd 'C(n,k)) <->
   exists i, n %% (2 ^ i) < k %% (2^i).
```

The sum of the elements of $\mathbf{T}_n$ is $3^n + 1$ (see next chapter). This can be restated as

$$\sum_{i<2^n} s_{i+2^n} = 3^n.$$

Let X_n be the quantity to be computed. Apply the relation (1.1) to X_{n+1}, and use $s(2k+2^{n+1}) = s(k+2^n)$ and $s(2k+1+2^{n+1}) = s(k+2^n)+s(k+1+2^n)$. Note that $\sum s(k+1+2^n) = \sum s(k+2^n)$ since $s(2^n) = 1$, so that $X_{n+1} = 3X_n$.

We show then

$$\sum_{p\le i<2p} \frac{s_i}{s_{i+1}} = (3p-1)/2, \qquad p = 2^n.$$

We use (1.1) again. We get $X_{n+1} = A_n + B_n$. Consider first B_n, case where the index is odd. Since $B_n = \sum s_{2i+1}/s_{2i+2} = \sum (s_i + s_{i+1})/s_{i+1} = \sum 1 + \sum s_i/s_{i+1}$, the value is found by induction. Now $A_n = \sum_i s_i/(s_i+s_{i+1})$. We split this sum in two, in the first part $p < i < p+p/2$, in the second part we replace $f(i)$ by $f(2p-i-1)$. If the generic term of the first part is $a/(a+b)$, by the palindrome condition, the generic term of the second part is $b/(b+a)$. The sum of these two terms is 1; it follows $A_n = p/2$ (note: if $n = 0$, this is $1/2$, in all other cases, it is an integer).

We have

$$\sum_{p\le i<2p} \frac{1}{s_i s_{i+1}} = 1, \qquad (p = 2^n).$$

(the sum of the i first terms is s_i/s_{p+i}, by virtue of (3.6)).

Let $F_i = s_i/s_{i+1}$. The relation $p \le i < 2p$ with $p = 2^n$ says that F_i is an element of the Stern-Brocot tree at depth n. We have shown: the sum of these terms is $(3p-1)/2$ (so that the mean value is asymptotically $3/2$). The simplicity of a rational number a/b is $1/(ab)$. So we have: the sum of the simplicities of the numbers of the Stern-Brocot table at fixed depth is equal to one.

```
Lemma fuscq_eq0 n: ((fusc n)%:Q == 0) = (n == 0)%N.
Lemma sum_fusc_row n: \sum_(2^n <= i < 2^n.+1) (fusc i) = 3 ^n.
Lemma sum_Stern1 n: \sum_(i<2^n) (Stern (i+2^n).*2) = (2^n)%N%:Q / 2%:Q.
Lemma sum_Stern n: \sum_(i<2^n) (Stern (i+2^n)) = ((3* 2^n)%N%:Q -1) / 2%:Q.
Lemma sum_Stern_simp n:
  \sum_(2^n<=i< 2^n.+1) 1/((fusc i) * (fusc i.+1))%N%:Q = 1%:Q.
```

3.6 An iterative formula

Dijkstra [6] proposes the following procedure to compute s_N. Initially, we have $n = N$, $a = 1$, $n = 0$. While n is non-zero, it is replaced by its half; in the even case a is replaced by $a+b$, in the odd case b is replaced by $b+a$; finally the return value is b.

We can rewrite this as: there is a function $F_n(a,b)$, such that

$$F_0(a,b) = b, \quad F_{2n}(a,b) = F_n(a+b,b), \quad F_{2n+1}(a,b) = F_n(a,b+a) \tag{3.14}$$

and $F_n(1,0) = s_n$. This function is nothing else than

$$F_n(a,b) = as_n + bs_{n+1}.$$

```
Definition Fusci n a b := a * (fusc n) + b * (fusc n.+1).

Lemma fusci_0 a b: Fusci 0 a b = b.
Lemma fusci_even n a b: Fusci (n.*2) a b = Fusci n (a + b) b.
Lemma fusci_odd n a b: Fusci (n.*2.+1) a b = Fusci n a (a + b).
Lemma fusci_val n :Fusci n 1 0 = fusc n.
Lemma fusci_vald n: Fusci n 1 1 = fusc n.*2.+1.
Lemma fusci_1 a b: Fusci 1 a b = a + b.
```

Dijkstra [6] expresses the palindrome condition (3.4) as: "the value of the function fusc does not change if we invert in the binary representation of the argument all "internal" digits; for instance $s_{19} = s_{29}$".

Define $A(u,v,w) = (2\cdot(2^v+u)+1)\cdot 2^w$. Let $p = 2^v$ and $q = 2^w$. Assume $u+u'+1 = p$ (this implies $u < 2^v$). The palindrome conditions $s_{p+u} = s_{p+u'+1}$ and $s_{p+u+1} = s_{p+u'}$ say $F_{p+u}(a,b) = F_{p+u'}(b,a)$. From $s_{2k+1} = F_k(1,1)$ one deduces $s_{A(u,v,w)} = s_{A(u',v,w)}$ when $u+u'+1 = 2^v$ (the result is obviously independent of w).

```
Lemma fusci_palindrome_aux u u' v a b:
   (u + u').+1 = 2^v ->
   Fusci (2 ^v + u) a b = Fusci (2^v + u') b a.
Lemma fusc_palindrome_bis u u' v w w'
   (aux := fun u v w => (2 ^v + u).*2.+1 *(2 ^w)):
   (u + u').+1 = 2 ^ v ->
   fusc (aux u v w)  = fusc (aux u' v w').
```

Note that any non-zero integer that is not a power of two can uniquely be written in the form $A(u,v,w)$ with $u < 2^v$. The binary representation of n is thus: a digit one, followed by some zeroes, the digits of u, a digit one and w digits zero. The total number of digits is $2+k+w$; the k digits between the two digits one are the *internal digits*. If we invert them, we get $n' = A(u',v,w)$, where u' is like u, with digits inverted.

We define here the procedure that inverts the internal digits: $i(n) = c'(C'(I(C(c(n)))))$, where $I(l)$ inverts the inner bits of l. Recall that the inner bits are those between the first and last "true", and that the last element of the list is "true"; our procedure modifies also this last bit, but this is harmless as C' ignores it.

```
Fixpoint bin_invert_aux L :=
  match L with
   | true :: L' => true :: [seq ~~ i | i <- L']
   | false :: L' => false ::  bin_invert_aux L'
   | nil => nil
end.

Definition bin_invert n :=
   nat_of_bin (num_of_list (bin_invert_aux (list_of_num (bin_of_nat n)))).
```

We have for instance $i(19) = 29$. More interesting are $i(0) = 0$ and $i(1) = 1$. We have $i(2n) = 2i(n)$ (outer bits are not modified). Let n be an integer, $p = 2^{(\log_2 n)-1}$. The condition $p < n$ says that n is neither zero nor a power of two. In this case, we have $n + i(n) = 3p$. The palindrome condition can be rewritten as $s_n = s_{i(n)}$. Since n and $i(n)$ have the same base two logarithm, it follows that i is involutive, i.e., $i(i(n)) = n$.

```
Lemma bin_invert_19:  (bin_invert 19) = 29.
Lemma bin_invert_0:  (bin_invert 0) = 0.
Lemma bin_invert_1:  (bin_invert 1) = 1.
Lemma log2_bin_invert n: log2(bin_invert n) = log2 n.
Lemma bin_invert_odd n: odd n -> odd(bin_invert n).
Lemma bin_invert_double n:  (bin_invert n.*2) =  (bin_invert n).*2.
Lemma bin_invert_exp2n n: (bin_invert 2^n) = 2^n.
Lemma bin_invert_p1 n: 2^((log2 n).-1) < n ->
  n + (bin_invert n) = 3* 2^ ((log2 n).-1).
Lemma bin_invertI n: bin_invert (bin_invert n) = n.
Lemma fusc_bin_invert n: fusc (bin_invert n) = fusc n.
```

We consider here Dijkstra's approach. We first have to correct the definition of I, by replacing the last bit by true. This does not alter the value of $i(n)$. Given a non-zero integer n and its binary expansion $C(c(n))$, there are two cases: either there is a single one at the end, and this is a fix-point of I, or it starts with k zeros, followed by a one, then a list l and a final one, and I inverts the the elements of l.

```
Definition bin_invert_alt L:=
  let l := bin_invert_aux L in
  if l is a :: l' then rcons (belast a l') true else l.

Lemma bin_invert_altE l a s: bin_invert_aux l = rcons s a ->
   bin_invert_alt l = rcons s true.
Lemma bin_invert_alt_compat n:
  bin_invert n =
  nat_of_bin (num_of_list (bin_invert_alt (list_of_num (bin_of_nat n)))).
Lemma bin_invert_correctA n l:
  bin_invert_alt (ncons n false (true:: (rcons l true))) =
   (ncons n false (true:: (rcons [seq ~~ i | i <- l] true))).
Lemma bin_invert_correctB n (L:= ncons n false [::true]):
  bin_invert_alt  L = L.
Lemma aux_for_bin_invert n (L := list_of_num (bin_of_nat n.+1)):
  (exists k, L = ncons k false [::true]) \/
  (exists k l, L= ncons k false (true:: (rcons l true))).
```

We consider F as a function of $C(c(n))$ rather than n. This function satisfies $F_{0::l}(a,b) = F_l(a+b,b)$, $F_{1::l}(a,b) = F_l(a,a+b)$ and $F_l(1,0) = s(c'(C'(l)))$. So $s_n = F_l(1,0)$ for $l = l = C(c(n))$ and $s_{i(n)} = F_{l'}(1,0)$ for $l' = C(c(i(n)))$. The objective is to prove that these quantities are equal. With the modified version of I we have $l' = I(l)$. The case where l has a single one is trivial as $l = I(l)$. Otherwise l starts with some zeros, has a one, is followed by s, then by a one, and $I(l)$ has the same form, with $\bar{s}$ instead of s, this is like s with the bits exchanged. We may ignore the initial zero bits at it replaces the first argument 1 by $1+0$. The first one replaces the second argument 0 by $0+1$. The result follows by $F_{s1}(a,b) = F_{\bar{s}1}(b,a)$ instantiated with $a = b = 1$, by induction on s: if s is empty the common value is $a+b$; otherwise a 1 in s replaces b by $a+b$ in the RHS; and the corresponding 0 is $\bar{s}$ produces the same effect in the LHS; $a'+b'$, but these quantities are the same.

```
Definition FusciL l a b := Fusci (nat_of_list l) a b.

Lemma fusciL_0 a b: FusciL nil a b = b.
Lemma fusciL_even l a b: FusciL (false::l) a b = FusciL l (a + b) b.
Lemma fusciL_odd l a b: FusciL (true::l) a b = FusciL l a (a + b).
Lemma fusciL_val l:
   FusciL (rcons l true) 1 0 = fusc (nat_of_list (rcons l true)).
Lemma fusciL_valn n: FusciL (list_of_num (bin_of_nat n)) 1 0 = fusc n.
Lemma fusc_bin_inverse_dijkstra n: fusc (bin_invert n.+1) = (fusc n.+1).
```

Dijkstra [6] says: "The next property is more surprising. (At least, I think so.) Let us try to represent the pair (a,b) by the single value m, according to $a = s_{m+1}$ and $b = s_m$". So define $G_n(m) = F_n(s_{m+1}, s_m)$. We obviously have

$$G_0(m) = s_m, \qquad G_{2n}(m) = G_n(2m), \qquad G_{2n+1}(m) = G_n(2m+1) \tag{3.15}$$

and $G_n(0) = s_n$.

```
Definition Fuscj n m := Fusci n (fusc m.+1) (fusc m).

Lemma fuscj_0 m:  Fuscj 0 m = fusc m.
Lemma fuscj_val n: Fuscj n 0 = fusc n.
Lemma fuscj_even n m: Fuscj (n.*2) m = Fuscj n (m.*2).
Lemma fuscj_odd n m: Fuscj (n.*2.+1) m = Fuscj n (m.*2.+1).
Lemma fuscj_1 n: Fuscj 1 n = Fuscj 0 (n.*2.+1).
```

Dijkstra concludes "the fusc-value does not change if we write the binary digits of the argument in reverse order". By fusc-induction on b it follows that $G_a(b) = G_0(a2^{\log_2 b} + r_2(b))$; it suffices then to take $a = 0$. Example: from $G_{19}(0) = G_0(25)$ we deduce $s_{19} = s_{25}$.

```
Lemma fuscj_reverse n: Fuscj n 0 = Fuscj 0 (base2rev n).
Lemma fusc_reverse n: fusc (base2rev n) = fusc n.
```

Let's do some obfuscation: two integers are coprime if and only if they are the fusc-value of indices who sum up to a power of two. The key relation is $r_2(2k+1) = 2^{\log_2(k)} + r_2(k)$. It implies, for odd n,

$$r_2(n-1) + i(r_2(n)) = 2^{\log_2 n}. \tag{3.16}$$

Let a and b be two coprime integers; we may assume $a \le b$, so that there is an odd integer n with $a/b = s_{n-1}/s_n$. If $u = r_2(n-1)$ and $v = i(r_2(n))$ then $a = s_{n-1} = s_u$ and $b = s_n = s_v$; the result follows from (3.16). Conversely, assume that $u+v$ is a power of two, and let's show that s_u and s_v are coprime. We may assume $u < v$, and, after division by a power of two, that v is odd. Now u even says $u + v = 1$, this case is trivial. So both numbers are odd, and there is k such that $u + v = 2^{k+1}$. Let $n = r_2(i(v))$; we have $v = i(r_2(n))$ and $s_v = s_n$. The relation $u < v$ says $u < 2^k \le v$, so that $\log_2 v = k+1 = \log_2 n$. Now, equation (3.16) says $r_2(n-1) = u$, so that $s_u = s_{n-1}$. It follows that s_u and s_v are coprime.

By the way, $u < v$ implies $s_u < s_v$ (proof by induction).

```
Lemma fusc_fusc1 n: odd n -> n != 1 ->
  (base2rev n.-1) + bin_invert(base2rev n) = 2^(log2 n).
Lemma fusc_fusc2 a b: coprime a b ->
   exists n m p, [/\ n + m = 2^p, a = fusc n & b = fusc m].
Lemma fusc_fusc3 n m p: n + m = 2^p -> coprime (fusc n) (fusc m).
Lemma fusc_compare1 i j k: i < j -> i + j = 2 ^ k -> fusc i < fusc j.
```

3.7 Weak base two decomposition

Let's show: the number of decompositions $S(n)$ of n as $\sum a_i 2^i$, where a_i can be 0, 1 or 2 equal to s_{n+1}. (for a similar proof, without padding, see [7]).

In what follows, we shall denote by l the sequence of the a_i, and by a an element of l. We replace the condition that a is 0, 1, or 2 by a type condition: a must be in I_3, and we define the sum $v(l) = \sum l_i 2^i$ by induction. Note that $v(l++l') = v(l) + 2^m v(l')$, where m is the size of l. If $v(l') = 0$ then all elements of l' must be zero. This is in particular the case when the size of l is $\ge log_2(n)$ with $n = v(l++l')$. Conversely, $v(l++l') = v(l)$ when all elements of l' are zero. We express

```
Section WeakBaseTwoRepresentation.
```

```
Fixpoint wbase2 (l: seq 'I_3) :=
  if l is a :: l' then a + (wbase2 l').*2 else 0.
Definition ord1_3:= (Ordinal (isT:1<3)).
Definition ord2_3:= (Ordinal (isT:2<3)).
Definition bool_to_I3 (a: bool) := if a then ord1_3 else ord0.

Lemma Ord3_trichot (a: 'I_3): (a == ord0) (+) (a == ord1_3) (+) (a == ord2_3).
Lemma wbase2_rec_nat (a:nat) l (H: a < 3):
  wbase2 (Ordinal H :: l) = a + (wbase2 l).*2.
Lemma wbase2_recr a l: wbase2 (rcons l a) = wbase2 l + 2 ^ size l * a.
Lemma wbase2_cat l1 l2:
  wbase2 (l1 ++ l2) =  wbase2 l1 + (2^(size l1)) * (wbase2 l2).
Lemma wbase2_zero l : wbase2 l = 0 -> l = nseq (size l) ord0.
Lemma wbase2_pad l k: wbase2 (l ++ nseq k ord0) = wbase2 l.
Lemma wbase2_split l n: log2 n <= size l -> wbase2 l = n ->
  l = take (log2 n) l ++ nseq ((size l) - log2 n) ord0.
```

We have $v(\mathrm{C}(c(n))) = n$, where $\mathrm{C}(c(n))$ is the binary decomposition of n, where true and false are converted to 1 and 0 in I_3, respectively. Obviously, $v(l)$ is odd when $l_0 = 1$, and conversely. So, if $v(l) = 2n+1$, then $l = 1 :: l'$ and $v(l) = n'$. If $l = v(2n+2)$ then l is non-empty, say $l = a :: l'$ where a is 0 or 2 and $v(l') = n+1$ or $v(l') = n$ respectively.

```
Lemma wbase2_correct n:
  wbase2 [seq bool_to_I3 i | i<- (list_of_num (bin_of_nat n))] = n.

Lemma wbase2_odd l : odd (wbase2 l) <-> head ord0 l = ord1_3.
Lemma wbase2_odd1 n l :
  wbase2 l = n.*2.+1 -> head ord0 l = ord1_3 /\ wbase2 (behead l) = n.
Lemma wbase2_double_nz n l (a := head ord0 l) (l' := behead l):
  wbase2 l = n.*2.+2 -> l = a :: l' /\
   ((a == ord0) && (wbase2 l' == n.+1) (+) ((a == ord2_3)&& (wbase2 l' == n))).
```

The difficulty is that we cannot consider the set of all lists. For this reason, we consider tuples (list of fixed length), obtained by padding with zeroes. If l is a list of size equal to k, it can be considered as a tuple (of some size) then cast into a k-tuple; the cast does not change the value (this property is not in the SSREFLECT library for some reason). If $n \leq k$ a n-tuple can be padded into a k-tuple and the v-value is left unchanged. The padding is injective.

```
Definition tuple_padl n k (l: n.-tuple 'I_3) :=
   (l ++ (nseq (k - n) ord0)).

Lemma tcast_val (T: Type) m n (H: m = n) (l: m.-tuple T):
  tval (tcast H l) = tval l.
Fact size_tuple_padl  n k (l: n.-tuple 'I_3): n <= k
  -> size (tuple_padl k l) = k.

Definition tuple_pad k n (l:n.-tuple 'I_3) (H: n <= k) :=
  (tcast (size_tuple_padl l H) (in_tuple (tuple_padl k l))).

Lemma tuple_pad_val k n (l:n.-tuple 'I_3) (H: n <= k):
  tval (tuple_pad l H) = tval l ++  (nseq (k - n) ord0).

Lemma wbase2_tuple_pad k n (l:n.-tuple 'I_3) (H: n <= k):
  wbase2 (tuple_pad l H) = wbase2 l.
```

```
Lemma tuple_pad_injective k n (H: n <= k):
  injective (tuple_pad (n:=n) ^~ H).
```

We define $S_k(n)$ to be the cardinal of the set $E_k(n)$ formed of all k-tuple t such that $v(t) = n$, and $S(n)$ the value for $k = \log_2 n$. We have $S_k(n) = S(n)$ when $\log_2 n \leq k$ (because padding is injective, and every element of $E_k(n)$ if of the form $l++l'$ where l has size $\log_2 n$ and elements of l' are zero. We have $E(0) = 1$, since $E(0) = E_0(0)$ and the empty sequence satisfies $v(t) = n$. We have $E(2n+1) = E(n)$ because $log_2(2n+1) = 1+log_2 n$, and as mentioned above $v(t) = 2n+1$ says that the first element of t is 1. We have $E(2n+2) = E(n)+E(n+1)$ because, as mentioned above, $v(t) = 2n+2$ says that the head of t is 0 or 2; we have $log_2(2n+2) = 1+log_2(n+1)$, and $log_2 n \leq log_2(n+1)$. The result follows by induction.

```
Definition card_wbase2' k n := #|[set t :k.-tuple 'I_3 | wbase2 t == n ]|.
Definition card_wbase2 n := card_wbase2' (log2 n) n.

Lemma card_wbase2_invariant k n : log2 n <= k ->
  card_wbase2 n = card_wbase2' k n.
Lemma card_wbase2_correct k1 k2 n:
   log2 n <= k1 -> log2 n <= k2 -> card_wbase2' k1 n = card_wbase2' k2 n.
Lemma card_wbase2_odd n: card_wbase2 n.*2.+1 = card_wbase2 n.
Lemma card_wbase2_even n:
  card_wbase2 n.*2.+2 = card_wbase2 n +  card_wbase2 n.+1.
Lemma card_wbase2_val n: card_wbase2 n = fusc n.+1.

End  WeakBaseTwoRepresentation.
```

Alternative Definition. There is an alternative definition, by induction on the binary representation of n. This provides the pair (s_n, s_{n+1})

```
Fixpoint fuscP2 p :=
   match p with
   | xO n => let: (a,b) := fuscP2 n in (a, a+b)
   | xI n => let: (a,b) := fuscP2 n in (a+b,b)
   | xH => (1,1)
end.
Definition fuscN2 p := if p is (Npos p) then (fuscP2 p) else (0,1).

Lemma fuscN2_prop n:
  fuscN2 (bin_of_nat n) = (fusc n, fusc n.+1).
Lemma fuscN_prop n: fusc n = (fuscN2 (bin_of_nat n)).1.
```

3.8 The Stern-Brocot tree

A tree is a data structure formed of nodes, related by a relation that reads "a is a child of b" or "b is a parent of a", that satisfies the following properties: a node may have several children, but only one parent; the transitive closure of the relation is antisymmetric (there is no cycle); the reflexive symmetric and transitive closure of the relation relates every node to every other node (whatever partition of the nodes in two non-empty sets A and B, at least one element of A is related to one element of B).

In a binary tree, each node has zero or two children. A node that has no parent is called a root, a node that has no child is called a leaf. A tree has at most one root; if there is a root,

there is, for each node, a unique finite path from the root to the node, the length of which is called the depth of the node, and one can prove properties by induction on the depth. Moreover, one can define an order on the tree: it is the only order such that if $\mathrm{L}(x)$ and $\mathrm{R}(x)$ are the left and right children of x, then $\mathrm{L}(x) < x < \mathrm{R}(x)$. There is a variant: $p \ll q$ if either the depth of p is less than the depth of q, and $p < q$ and the depths are the same.

Consider the following example from [1, section 6.3.4]:

```
Inductive Z_btree : Set :=
  Z_leaf : Z_btree
| Z_bnode : Z -> Z_btree -> Z_btree -> Z_btree.
```

Here a node is either a leaf, or a parent of two nodes and has a label (of type Z). In order to understand the structure of the tree, let's denote the leaf by 0, and '`Z_bnode x y z`' by $y+z$ (this notation hides the label). The nodes are then 0, $0+0$, $0+(0+0)$, $(0+0)+0$, $(0+0)+(0+0)$, etc.; they are at depth 0, 1, 2, 2 and 2 respectively. This is a complicated data structure, and does not respect the rules given above. We have in mind something easier. For instance, we consider a root denoted 1, and every node x has two children, denoted $x0$ and $x1$. Some nodes are 1, 10, 11, 101, 111, 100, 110 (at depth 0, 1, 1, 2, 2, 2, and 2 respectively). One recognizes the non-zero integers, written in base two, so that every binary tree, in this restricted form, is isomorphic to:

```
Inductive positive : Set :=
  | xH : positive
  | xI : positive -> positive
  | xO : positive -> positive.
```

One can associate a label to each node; the obvious example is its numeric value (this is called $c'(x)$ elsewhere). In the first chapter, we have associated a list of bits (in the reverse order of the digits given above). Another example is the following:

```
Fixpoint SB_to_rat x :=
  match x with xH => 1%Q
     | xO x' => Sba_i (SB_to_rat x')
     | xI x' => Sba_p (SB_to_rat x')
  end.
```

The two functions used in this definition are G_i and G_p, they are called N and D in [1, Exercise 6.24] because $\mathrm{G}_i(a/b) = \mathrm{D}(a/b) = a/(a+b)$ and $\mathrm{G}_p(a/b) = \mathrm{N}(a/b) = (a+b)/b$. Of the node p is interpreted as the integer $n+1$ this function produces $\mathrm{F}_n = s_n/s_{n+1}$.

```
Lemma SB_to_rat_val n:
  SB_to_rat (pos_of_nat' n) = Stern (n.+1).
\end{verbatim
Here is another example, \cite{bertat-niqui}
```

Here is another example, from [9]. The tree is isomorphic to the previous one, and the labels are the numerators and denominators of F_n (hence are s_n and s_{n+1}). This is a possible implementation of the rational numbers, moreover, the four operations can be expressed on this representation (see [8]).

```
Inductive Qpositive : Set :=
  | nR : Qpositive -> Qpositive
```

```
  | dL : Qpositive -> Qpositive
  | One : Qpositive.

Fixpoint Qpositive_i (w : Qpositive) : nat * nat :=
  match w with
  | One => (1, 1)
  | nR w' => match Qpositive_i w' with
             | (p, q) => (p + q, q)
             end
  | dL w' => match Qpositive_i w' with
             | (p, q) => (p, p + q)
             end
  end.
```

What we get is a labelled tree, where each rational number $x > 0$ appears exactly once as a label. This tree is called the Calkin-Wilf tree. Let $c(p)$ be the label of node p. Note that $c(\mathrm{L}x) < 1 < c(\mathrm{R}x)$, so that the labelling does not respect the order.

The Stern-Brocot tree has the same set of labels for a given level, in increasing order. Let $b(p)$ be the label of node. We have $b() = 1/1$, $b(\mathrm{L}) = 1/2$, $b(\mathrm{R}) = 2/1$, $b(\mathrm{LL}) = 1/3$, $b(\mathrm{LR}) = 2/3$, $b(\mathrm{LR}) = 3/2$, $b(\mathrm{RR}) = 3/1$ (the labels for LR and RL are swapped). We have also $b(\mathrm{LRL}) = 3/5$ and $b(\mathrm{LRLL}) = 4/7$. So one goes from $1/1$ to $4/7$ via $1/2$, $2/3$ and $3/5$. Consider the sequence $4/7$, $4/3$, $1/3$, $1/2$, $1/1$ (in each case we subtract the numerator from the denominator or vice versa, if we call these operations L and R, we have performed, in order LRLL). More generally, if q is the dual node of p (obtained by inverting the sequence of L and R), then $b(p) = c(q)$. Computing $b(p)$ is a bit tricky, as well as obtaining the sequence $1/2$, $2/3$ and $3/5$ from $4/7$.

Let's first consider the ordering of the tree: everything to the left of a node is less than everything to the right, and the node is between them. We can restate this with two rules: whatever x, y, z, $xz < yz$ is equivalent to $x < y$ and $x\mathrm{L} < \emptyset < y\mathrm{R}$. So to compare two nodes, one removes the common suffix, and checks the rightmost value. For instance $110 < 1101 < 11$ (in base ten, this reads $6 < 13 < 3$); this is the same as $\mathrm{RL} < \mathrm{LRL} < \mathrm{R}$.

Consider now a node p at depth n. Let E be the set of nodes at depth $< n$, $\mathrm{E}_<$ those in E that are $< p$ and $\mathrm{E}_>$ those that are $> p$. Let $q = \max \mathrm{E}_<$ and $q' = \min \mathrm{E}_>$ (use pseudo nodes $-\infty$ and $+\infty$ in case of empty sets). These two nodes are called the left and right neighbor of p. Assume $n > 0$ (this excludes the root that has no neighbor). It is clear that $\mathrm{L}^{n-1)}$ is the smallest element in E, and $\mathrm{R}^{n-1)}$ is the largest. Moreover, unless $p = \mathrm{L}^n$, we have $\mathrm{L}^{n-1} < p$, and p has a left neighbor. Assume, for instance $p = \mathrm{L}q$; in this case $p < q$. Assume $p < x < q$. Remove the common right prefix of x and q. We get $\mathrm{L}q' < x' < q'$ for some x' and q'. This equation is absurd if q' is not empty; otherwise it says that x' has the form $x''\mathrm{RL}$, so x is $x''\mathrm{R}p$; this is absurd as its says that x has depth $> n$. So: the right neighbor of the left child of q is q; and we may exchange left and right in this sentence.

With the notations above, $c(p) = c(q) \oplus c(q')$, where $x \oplus y$ is some magic function. One can interpret $c(p)$ as a pair of integers. In this case $c(-\infty) = (1,0)$, $c(+\infty) = (0,1)$ and $(a,b) \oplus (c,d) = (a+c, b+d)$. Since the root has no real neighbor, its label is $(1,1)$. It is easy to show, by induction that $c(\mathrm{L}^n) = (1, n+1)$ and $c(\mathrm{R}^n) = (n+1, 1)$. It follows that $c(p)$ is always a pair of non-zero integers. So, one could consider $c(p)$ as a rational number and define the magic function by $a/b \oplus c/d = (a+c)/(b+d)$.

One objective is to show that each rational number is exactly once in the tree. This can be restated: every pair (a,b), where $a > 0$, $b > 0$, and $a \perp b$ is uniquely of the form $c(p)$. But we also want to show that $p < q$ implies $c(p) < c(q)$. If $\alpha = (a,b)$, $\beta = (c,d)$ and $\mathrm{A}(\alpha,\beta) = bc - ad$, then $\mathrm{A}(\alpha,\beta) > 0$ means $\alpha < \beta$, when α and β are considered as rational numbers. It happens

that this function A is the key to the termination of the algorithm.

We start our implementation with pairs of integers. The pair $(0,0)$ will be called NaN (not a number) as it cannot be interpreted as a rational number. We say that $a = (a_1, a_2)$ is good if a_1 and a_2 are coprime; this implies it is not NaN and can be interpreted as a rational number in the range $[0, +\infty]$. We say that it is very good if moreover none of a_1 and a_2 is zero so that a can be interpreted as a rational number in the range $]0, +\infty[$. We equip this set of pairs with an order compatible with the order on **Q** induced by interpreting a pair as a rational number.

Note: in order to use $a < b$ as notation for comparison of pairs, we construct and open a new scope; The current scope is ring-scope, so that it will be often necessary to specify that certain operations must be interpreted in the nat scope.

```
Definition pair_le a b:= (a.1 * b.2 <= a.2 * b.1)%N.
Definition pair_lt a b:= (a.1 * b.2 < a.2 * b.1)%N.
Definition good_pair a:= coprime a.1 a.2.
Definition very_good_pair a:= [&& good_pair a, 0< a.1 & 0< a.2]%N.
Definition NaN := (0,0)%N.

Notation "a <= b" := (pair_le a b): nat_pair_scope.
Notation "a < b" := (pair_lt a b): nat_pair_scope.
```

The two relations $\leq$ and $<$ have some surprising properties when one argument is NaN.

```
Lemma good_pair_prop a b:
  good_pair a -> good_pair b -> (a.1 * b.2 = a.2 * b.1)%N -> a = b.

Lemma pair_le_bad a : (a <= NaN) && (NaN <= a).
Lemma pair_le_bad' a:  ~~(a < NaN) && ~~(NaN < a).
Lemma pair_le_nn a: a <= a.
Lemma pair_lt_nn a: (a < a) = false.
Lemma pair_le_anti a b: good_pair a -> good_pair b ->
  ((a <= b) && (b <= a)) = (a == b).
Lemma pair_le_trans n m p: n != NaN ->
   m <= n -> n <= p -> m <= p.
Lemma pair_le_lt_trans n m p: m != NaN ->
   m <= n -> n < p -> m < p.
Lemma pair_lt_le_trans n m p: p != NaN ->
   m < n -> n <= p -> m < p.
Lemma pair_lt_trans n m p: m < n -> n < p -> m < p.
Lemma pair_lt_le a b:  good_pair a -> good_pair b ->
  a < b = (a <= b) && (a != b).
Lemma pair_lt_trichot a b:  good_pair a -> good_pair b ->
  (a < b) (+) (b < a) (+) (a == b).
```

We introduce now the function $A(a,b)$. Note that A takes its values in **Z**; and $a \leq b$ is equivalent to $0 \leq A(a,b)$. In the case of good pairs, $A(a,b) = 0$ is equivalent to $a = b$. Let's say that a and b are near when $A(a,b) = 1$; this implies $a < b$; by abuse of language we also say that b and a are near. As $A(a,b) = 1$ is a Bezout relation, it implies that near pairs are good.

```
Definition area a b := (a.2)%:Z * (b.1)%:Z - (a.1)%:Z * (b.2)%:Z.
Definition pair_compat a b := area a b == 1.

Lemma area_xx a : area a a = 0.
Lemma area_sym a b : area a b = - (area b a).
```

```
Lemma area_gt0 a b: (a < b) = (0 < area a b)%R.
Lemma area_ge0 a b: (a <= b) = (0 <= area a b)%R.
Lemma area_eq0 a b: good_pair a -> good_pair b ->
   (area a b) == 0 -> a = b.

Lemma pair_nearE a b: (pair_near a b) = (a.2 * b.1 == a.1 * b.2 +1)%N.
Lemma pair_near_good a b: pair_near a b -> good_pair a && good_pair b.
```

We introduce now the operation $a \oplus b$. This operation is commutative, and compatible with the order. This means that $a \leq b$ implies $a \leq a \oplus b \leq b$. We have a distributivity relation $\mathrm{A}(a, b \oplus b') = \mathrm{A}(a,b) + \mathrm{A}(a,b')$. If a and b near, then $a \oplus b$ is near to a and b (in particular good).

```
Definition pair_add a b := (a.1 + b.1, a.2 + b.2)%N.
Notation "a + b" := (pair_add a b): nat_pair_scope.

Lemma pair_addC a b: a + b = b + a.
Lemma pair_add_monotone a b: a <= b -> (a <= a + b) && (a + b <= b).
Lemma pair_add_smonotone a b: a < b -> (a < a + b) && (a + b < b).
Lemma area_Dr a b b': area a (b+b') = ((area a b) + (area a b'))%R.
Lemma area_Dl a a' b: area (a+a') b = ((area a b) + (area a' b))%R.
Lemma add_near_l a b: pair_near a b -> pair_near a (a+b).
Lemma add_near_r a b: pair_near a b -> pair_near (a+b) b.
Lemma add_near_g a b: pair_near a b -> good_pair (a+b).
```

Define the distance from x to a pair a, b be $|\mathrm{A}(a,x)| + |\mathrm{A}(x,b)|$. This is a natural number, and simplifies to $\mathrm{A}(a,x) + \mathrm{A}(x,b)$ when $a < x < b$. Assume moreover $x < a + b$. Then the distance from x to a, $a + b$ is $\mathrm{A}(x,b)$. In particular it is smaller than the distance to a, b.

```
Definition areaN a b := (a.2 * b.1 - a.1 * b.2) %N.
Definition pair_dist x a b:= (areaN a x + areaN x b) %N.

Lemma pair_distl x a b: a < b -> a < x -> x < a + b ->
  (pair_dist x a (a+b)%nat_pair < pair_dist x a b) %N.

Lemma pair_distl x a b: a < b -> a < x -> x < a + b ->
   pair_dist x a (a+b) = areaN x b.
Lemma pair_distr x a b: a < b -> a + b < x -> x < b ->
   pair_dist x (a+b) b = areaN a x.
```

Consider the following algorithm. The input is x, a very good pair. There are three variables, a, b, p, initialized to $(0,1)$, $(1,0)$ and the root of the tree, respectively. Initially $a < x < b$ since x is very good; moreover a is near b. This will be the invariant. Let's compare x with $a \oplus b$. It is less than, greater than or equal to it. In case of equality, the algorithm stops, and returns p. Otherwise its replaces p by its left (respectively right) child and b (respectively a) by $a \oplus b$. Obviously, the invariant is preserved, and the distance from x to a, b is strictly increasing.

This returns, for each strictly positive rational number x its position in the Stern Brocot tree.

```
Fixpoint SB_rec n x a b p :=
    if n is n'.+1 then
```

```
     if (x < a + b) then SB_rec n' x a (a+b) (xO p)
     else if (a + b < x) then SB_rec n' x (a+b) b (xI p)
     else p
    else p.

Definition SB_val x := let a := (0%N,1%N) in let b := (1%N,0%N) in
   SB_rec (pair_dist x a b) x a b xH.
```

Let $p(x)$ be this function. What remains to do is formalize the ordering on the tree, show that $x < x'$ implies $p(x) < p(x')$, and that every node has the form $p(x)$. After that, one can prove that this tree is the dual of the Calkin-Wilf tree.

Chapter 4

The Stern diatomic sequence

Let R be the function that maps the list $[u, v, w, \ldots]$ onto the list $[u, u+v, v, v+w, w, \ldots]$. For completeness, it is the identity for the empty lists and singletons. Otherwise, it maps a list of size $p+1$ onto a list of size $2p+1$. Elements at even position of R(L) are called "Stammglied" by Stern in [10], they appear in L; elements at odd position are called "Summenglied", they are sums of consecutive elements of L. The element at position p is called the "middle" element; The first and second half of the list are formed of elements with index $\leq p$ or $\geq p$ (this means that the middle element belongs to both parts, but it could be excluded as well).

If we start with $[u, v]$ and iterate n times, we get a list denoted here $\mathrm{S}_n(u, v)$. The set of all $\mathrm{S}_n(u, v)$ is called the Stern diatomic the table (Stern says "Entwickelung") and each $\mathrm{S}_n(u, v)$ is a row (Stern says "Entwickelungsreihe"). The i-th element of the list (or row) is denoted by $\mathrm{S}_n(u, v)(i)$, or $\mathrm{S}_n(i)$ when u and v are obvious from the context. We have $\mathrm{S}_0(u, v)(0) = u$ and $\mathrm{S}_0(u, v)(1) = v$. The quantities u and v are the "Arguments" of the table.

The two quantities $\mathbf{A} = \mathrm{S}(0, 1)$ and $\mathbf{B} = \mathrm{S}(1, 1)$ are very interesting. First, $\mathbf{A}_k(i)$ is independent of k when k is large enough; let's denote this number by s_i. This is the "Stern sequence", it is identical to the fusc function introduced in the previous chapter. Second, the first and last elements of $\mathbf{B}_k$ are equal to 1; if we merge all these $\mathbf{B}_k$ into a single list, removing the final 1 of each row, we get the sequence s_n (minus its leading zero).

We shall denote by $\mathrm{C}(a, b)$ the property that a and b are consecutive in some row of $\mathbf{B}$. This property is equivalent to $a \perp b$ (i.e., a and b are coprime). If this holds then $\mathrm{S}(a, b)$ is a subtree of $\mathbf{B}$, in the following sense: assume that $a = \mathbf{B}_k(i)$ and $b = \mathbf{B}_k(i+1)$. Then, for every q and j, we have $\mathrm{S}_q(a, b)(j) = \mathbf{B}_{k+q}(2^k i + j)$. More generally, if c is the gcd of a and b, there are numbers i and k such that $a = c\mathbf{B}_k(i)$ and $b = c\mathbf{B}_k(i+1)$. Therefore $\mathrm{S}_q(a, b)(j) = c\mathbf{B}_{k+q}(2^k i + j)$.

4.1 Definitions

Let's start with some definitions. Note that $\mathrm{A} \pm \mathrm{B}$ is the list whose element at position i is $\mathrm{A}(i) \pm \mathrm{B}(i)$ provided that both terms are defined, and $c \cdot \mathrm{A}$ is the list of all $c \cdot \mathrm{A}(i)$.

```
Fixpoint sternd_rec s : seq nat :=
  if s is a :: s1 then if s1 is  b :: _ then [:: a, a+b & sternd_rec s1]
  else  [:: a] else nil.

Fixpoint sternD (u v n: nat) :=
```

```
  if n is k.+1 then sternd_rec (sternD u v k) else  [:: u; v].
Definition stern11 := sternD 1 1.
Definition stern01 := sternD 0 1.
Definition stern i := nth 0 (stern01 i) i.

Definition add_seq s1 s2 := [seq x.1 + x.2 | x <- zip s1 s2].
Definition sub_seq s1 s2 := [seq x.1 - x.2 | x <- zip s1 s2].
Definition scale_seq k s := [seq k * x | x <- s].
```

We first consider some examples (equations (1), (2) and (3) of Stern). The quantity S_1 is the first row, S_2 is the second row (and S_0 is the zero-th row). The first column contains m, the second contains $km+n$, etc. More generally, all elements are of the form $km+ln$. The quantity k called the *first coefficient* and l called the *second coefficient.* It is obvious that k and l do not depend on m and n, but only on the position in the table. We shall see below that if the element is at row i, column j (starting the enumeration with zero), then $k = s_{2^i+j}$ and $l = s_j$ (see equation (4.6)).

$$S_1(m,n) = [m, m+n, n],$$

$$S_2(m,n) = [m, 2m+n, m+n, m+2n, n],$$

$$S_3(m,n) = [m, 3m+n, 2m+n; 3m+2n, m+n, 2m+3n, m+2n, m+3n, n].$$

```
Lemma stern_ex1 m n: sternD m n 1 = [:: m; m+n ; n].
Lemma stern_ex2 m n: sternD m n 2 = [:: m; 2 * m +n; m + n ; m + 2 *n; n].
Lemma stern_ex3 m n:
 sternD m n 3 =
 [:: m; 3*m + n; 2*m+n; 3*m+2*n; m+n;  2*m+3*n; m + 2*n; m+3*n ; n].
```

We start with obvious properties: $S_n(u,v)$ has size 2^n+1, starts with u and ends with v.

```
Lemma size_sternD u v n: size (sternD u v n) = (2^n).+1.
Lemma head_sternD u v n: head 0 (sternD u v n) = u.
Lemma last_sternD u v n: last 0 (sternD u v n) = v.
Lemma sternD_split1 u v n: sternD u v n = u :: behead (sternD u v n).
Lemma sternD_split2 u v n: sternD u v n =
   rcons (belast u (behead (sternD u v n))) b.
```

Let's denote by $r(\mathrm{L})$ the reverse of L. A careful analysis of R shows that $r(\mathrm{R}(\mathrm{L})) = \mathrm{R}(r(\mathrm{L}))$. In particular $r(\mathrm{S}(u,v)) = \mathrm{S}(v,u)$, and $\mathbf{B}$ is symmetric.

```
Lemma sternD_rec_aux a s:
  s != nil -> sternd_rec (a::s) = [:: a, (a + head 0 s) & sternd_rec s].
Lemma sternD_rev u v n:  sternD v u n = rev (sternD u v n).
Lemma stern11_symmetric n: rev (stern11 n) = (stern11 n).
```

Let's denote by $x \oplus y$ the operation that consists in concatenating x and y, without the head of y. If this head is the tail of x, we have $\mathrm{R}(x \oplus y) = \mathrm{R}(x) \oplus \mathrm{R}(y)$. It follows

$$\mathrm{S}_{n+1}(u,v) = \mathrm{S}_n(u, u+v) \oplus \mathrm{S}_n(u+v, v). \tag{4.1}$$

We can express this as: the list of the first (resp. last) 2^n+1 elements of $\mathrm{S}_{n+1}(u,v)$ is respectively $\mathrm{S}_n(u,u+v)$ and $\mathrm{S}_n(u+v,v)$. A special case is: $\mathbf{A}_{n+1} = \mathbf{A}_n \oplus \mathbf{B}_n$, or

$$\mathrm{S}_{n+1}(0,1) = \mathrm{S}_n(0,1) \oplus \mathrm{S}_n(1,1). \tag{4.2}$$

```
Notation "x ++- y" :=  (x ++ behead y) (at level 60).

Lemma split_sternD u v n :
  (sternD u v n.+1) = (sternD u (u + v) n) ++- (sternD (u + v) v n).
Lemma split_sternD_left u v n :
  take ((2^n).+1) (sternD u v n.+1) = sternD u (u + v) n.
Lemma split_sternD_right u v n :
  drop (2^n) (sternD u v n.+1) = sternD (u + v) v n.

Lemma split_stern01_left n : take ((2^n).+1) (stern01 n.+1) = stern01 n.
Lemma split_stern01_right n : drop (2^n) (stern01 n.+1) = stern11 n.
```

The characteristic property of R is that, if B = R(A), then $B_{2i} = A_i$ and $B_{2i+1} = A_i + A_{i+1}$. We deduce

$$B_{2i} = A_i, \quad B_{2i+1} = B_i + B_{i+1} \qquad A := S_n(u,v),\ B := S_{n+1}(u,v). \tag{4.3}$$

```
Lemma sternD_nth_even u v n k:
  nth 0 (sternD u v n.+1) (k.*2) = nth 0 (sternD u v n) k.
Lemma sternD_nth_odd u v n k:
  k < 2^n ->
  nth 0 (sternD u v n.+1) (k.*2.+1) =
  nth 0 (sternD u v n) k + nth 0 (sternD u v n) k.+1.
```

From (4.2) we deduce that $\mathbf{A}_n(i)$ is independent of n for large n. We denote this quantity by $s(i)$. The same formula says that $\mathbf{B}_n(i) = s(2^n + i)$. By symmetry we also have $\mathbf{B}_n(i) = s(2^{n+1} - i)$. From (4.3), it is clear that $s(n)$ satisfies the relation (3.1), thus is equal to the fusc function s_n. Since the condition $C(a,b)$ is equivalent to: there is an integer n such that $a = s(n)$ and $b = s(n+1)$, one has: this n is unique and $C(a,b)$ is equivalent to $a \perp b$. We shall see below how Stern proves these properties.

```
Lemma stern_prop i j: i <= 2^j ->  nth 0 (stern01 j) i = stern i.
Lemma nth_stern11 n i: i <= 2^n ->
    nth 0 (stern11 n) i = stern (2^n + i).
Lemma nth_stern11_sym n i: i <= 2 ^n ->
  nth 0 (stern11 n) i = stern (2^n.+1 - i).
Lemma stern_fusc: fusc_prop stern.
Lemma sternD_fusc: stern =1 fusc.
```

The Stern table is linear:

$$S_n(a,b) \pm S_n(c,d) = S_n(a \pm c, b \pm d), \tag{4.4}$$

so that $k \cdot S_n(a,b) = S_n(k \cdot a, k \cdot b)$. It follows equation (5) in Stern's paper:

$$S_n(1,2) - S_n(1,1) = S_n(0,1). \tag{4.5}$$

Since $S_n(1,2)$ is the first half of $S_{n+1}(1,1)$, it follows that $\mathbf{B}_{n+1}(i) - \mathbf{B}_n(i) = \mathbf{A}_n(i)$. Since $\mathbf{A}_n(i) = s_i$, it follows that, for fixed i, the i-th term of $\mathbf{B}_n$ forms an arithmetic progression, whose common difference is $s(i)$. This is the main result of §11 of Stern (compare with equation (3.5)).

```
Lemma scale_sternD k a b n:
  sternD (k*a) (k*b) n = scale_seq k (sternD a b n).
```

```
Lemma add_sternD a b c d n:
  sternD (a + c) (b + d) n = add_seq (sternD a b n) (sternD c d n).
Lemma sub_sternD a b c d n: c <= a -> d <= b ->
  sternD (a - c) (b - d) n = sub_seq (sternD a b n) (sternD c d n).
Lemma stern_equation5 n: sub_seq (sternD 1 2 n) (stern11 n) = stern01 n.
Lemma sternD_col_progression n i : i <= 2^n ->
  nth 0 (stern11 n.+1) i =  nth 0 (stern11 n) i + stern i.
```

A consequence of equation (4.4) is $S(u,v) = u \cdot S(1,0) + v \cdot S(0,1)$ so that

$$S_n(u,v)(i) = u \cdot s(2^n - i) + v \cdot s(i). \tag{4.6}$$

```
Lemma arith_split_sternD u v n:
   sternD u v n =
   add_seq (scale_seq u (rev (stern01 n))) (scale_seq v (stern01 n)).
Lemma sternD_stern u v n i: i <= 2^n ->
  nth 0 (sternD u v n) i = u * (stern (2^n - i)) + v * (stern i).
```

Relation (4.6) is very important. We give here some trivial consequences. First, it is obvious by induction that all terms in $S(0,1)$ are non-zero, except the first column. If follows $s(n) > 0$ when $n > 0$. Thus every element of $S(u,v)$ (except the first, equal to u, and the last, equal to v) is at least $u+v$.

```
Lemma sternD_01_positive n i:
  0 < nth 1 (sternD 0 1 n) i.+1.
Lemma stern_gt0 k: (0 < stern k) = (0 < k).
Lemma sternD_ge_in_bounds u v n i: 0 < i < 2^n ->
  u + v <= nth 0 (sternD u v n) i.
Lemma sternD_at_bounds u v n :
  nth 0 (sternD u v n) 0 = u /\ nth 0 (sternD u v n) (2^n) = v.
```

If we consider $u = v = 1$ in (4.6) and assume $i \le 2^n$, we get $\mathbf{B}_n(i) = s(2^n - i) + s(i)$. This is an alternative formula for $\mathbf{B}_n(i)$. Thus $s(i) \le \mathbf{B}_n(i)$ with equality only when $i = 2^n$. Stern deduces (in §11) that $\mathbf{B}_{n+1}(i) \le 2\mathbf{B}_n(i)$ with equality only when $i = 2^n$.

```
Lemma nth_stern11_alt n i: i <= 2^n ->
  nth 0 (stern11 n) i = stern (2^n - i) + stern i.
 Lemma sternD_bound1 i n: i <= 2 ^n ->
  stern i <= (nth 0 (stern11 n) i) ?= iff (i == 2 ^ n).
Lemma sternD_bound2 n i: i <= 2 ^ n ->
  nth 0 (stern11 n.+1) i <=  (nth 0 (stern11 n) i).*2 ?= iff (i==2^n).
```

If $u > 0$, $v > 0$, then all elements of $S(u,v)$ are > 0. The quantity $u+v$ is in the middle of each row and nowhere else.

```
Lemma sternD_positive u v n i : 0 < u -> 0 < v -> i <= 2^n ->
   0 < nth 0 (sternD u v n) i.
Lemma sternD_middle u v n : nth 0 (sternD u v n.+1) (2^n) = u + v.
Lemma sternD_middle_contra a b n i : u + v != 0 ->
   nth 0 (sternD u v n.+1) i = u + v -> i = 2 ^ n.
```

Stern says also: if a, b and c are three consecutive elements of $S(u,v)$ and if b is at odd position then b is greater than a and c (note that $b = a+c$). If by "greater than" one means $\ge$,

then the result is obvious; otherwise u and v have to be non-zero. As a and c are consecutive on the previous row, one of the terms is at odd position: it follows by induction, that if b is on row n, then $n < b$ (we shall use this property later on for **B**).

```
Lemma sternD_nth_compare u v n i (S := (sternD u v n.+1)):
  i < 2^n -> maxn (nth 0 S (i.*2)) (nth 0 S (i.*2.+2)) <= nth 0 S (i.*2.+1).
Lemma sternD_nth_compare_strict u v n i (S := (sternD u v n.+1)) :
   0 < u -> 0 < v -> i < 2^n ->
   maxn  (nth 0 S (i.*2)) (nth 0 S (i.*2.+2)) < nth 0 S (i.*2.+1).
Lemma sternD_nth_compare_rec u v n i:
   0 < u -> 0 < v -> i.*2 < 2^n ->
   n < nth 0 (sternD u v n) i.*2.+1.
```

In §2 Stern computes the sum $\mathbf{S}_p(u,v)$ of the terms of $\mathrm{S}_p(u,v)$. From (4.1) it follows

$$\mathbf{S}_{p+1}(u,v) = \mathbf{S}_p(u,u+v) + \mathbf{S}_p(u+v,v) - (u+v)$$

hence

$$\mathbf{S}_p(u,v) = \frac{3^p+1}{2}(u+v).$$

Stern notes that $\mathbf{S}_p(u,v)/\mathbf{S}_p(u',v')$ is independent of p and

$$\mathbf{S}_p(u+u',v+v') = \mathbf{S}_p(u,v) + \mathbf{S}_p(u',v').$$

```
Definition sternD_sum u v n := \sum_(i <- (sternD u v n)) i.

Lemma sternD_sumE u v n :  sternD_sum u v n = ((3^n).+1)./2 * (u + v).
Lemma sternD_sumC u v n : sternD_sum u v n = sternD_sum v u n.
Lemma sternD_sum_quo a b a' b' n :
  ((sternD_sum a b n) %:Q / (sternD_sum a' b' n) %:Q) %R =
   ((a +b )%N%:Q / ((a'+b')%N%:Q))%R .
Lemma sternD_sum_lin a b a' b' n :
  (sternD_sum a b n) +(sternD_sum a' b' n) =  sternD_sum (a+a') (b+b') n.
```

4.2 The case S(1,1)

Stern considers (in sections 3 through 12) the special case of $\mathrm{S}(1,1)$. He says: each row contains 1 at the beginning and end and nowhere else, it contains 2 in the middle and nowhere else, and is symmetric. This is obvious from the previous results.

```
Lemma head_sternD' n: head 0 (stern11 n) = 1.
Lemma last_sternD' n: last 0 (stern11 n) = 1.
Lemma SternD_positive' n k: 0 < k < 2^n -> 1 < nth 0 (stern11 n) k.
Lemma sternD_middle' n : nth 0 (stern11 n.+1) (2^n) = 2.
Lemma sternD_middle_contra' n k : nth 0 (stern11 n.+1) k = 2 -> k = 2 ^ n.
```

In §3, Stern wonders whether elements of the table can be even or odd. Let's write $\mathrm{C}(a,b)$ or $\mathrm{C}(a,b,c)$ when the two or three elements are consecutive in the same row. If $\mathrm{C}(a,b)$ holds, one of a or b has to be odd. For either $b = a+b'$ where b' follows b, or $a = a'+b$, where a precedes a', and one has respectively $\mathrm{C}(a,b')$ or $\mathrm{C}(a',a)$ in the previous row. If a and b are even so is b' or a'. The result follows by induction [replacing "even" by "multiple of p", the same argument says that a and b are coprime].

Let's says that (a,b,c) is of type 'oeo' if a is odd, b is even, c is odd; etc. If $C(a,b,c)$ then the cases 'eoe' and 'ooo' are excluded. This can be rephrased as: if b is odd, so is $a+c$. This is obvious when b is at an odd position (since $b=a+c$). Otherwise, the terms are $a'+b$, b, $b+c'$, with $C(a',b,c')$ on the previous row, and $a'+b+b+c'$ is $a'+c'$ modulo 2; the result is thus trivial by induction. Thus a triple can be only of type 'oeo' , 'eoo' or 'ooe'. By induction each row starts alternatively with 'oeo' and 'ooe'; this gives an easy way to find the parity of an element in the table. This can be restated as: s_n is even if and only if n is a multiple of three [this has been proved in the previous chapter].

In §4 Stern says: if $C(a,b,c)$ holds, then b divides $a+c$. Assume moreover that b is at position $i=(2q+1)\cdot 2^k$ on some row. If $k=0$, then i is odd and the result is obvious as $b=a+c$; otherwise, with the notations above $a+c=a'+b+b+c'$; moreover $C(a',b,c')$ holds on the previous row, where b is at position $i=(2q+1)\cdot 2^{k-1}$; the result holds by induction; moreover the quotient is $2k+1$. This is equation (4.) in Stern, and corresponds to equation (3.9) proved above. Note that the result holds for any $S(u,v)$. It follows: if b is odd, so is $a+c$. Note: one could easily generalize this as: a given element of $S(u,v)$ is odd if and only if either u and v are odd, and the corresponding element of $S(1,1)$ is odd, or u is odd, v is even, and the element is at the start of the row, or u is even, v is odd, and the element is at the end of the row.

```
Lemma sternD_middle_quo u v k q n (i:= 2^k * (q.*2.+1)) (S := sternD u v n):
  i < 2^n -> nth 0 S i.-1 + nth 0 S i.+1 = (k.*2.+1) * (nth 0 S i).
Lemma sternD_middle_div u v i n  (S := sternD u v n):
  0 < i < 2^n -> (nth 0 S i) %|  (nth 0 S i.-1 + nth 0 S i.+1).
Lemma sternD_odd_rec u v k q n (i:= 2^k * (q.*2.+1)) (S := sternD u v n):
  i < 2^n -> odd (nth 0 S i) ->  odd(nth 0 S i.-1 + nth 0 S i.+1).
```

In §5 Stern says that two consecutive numbers a and b have to be coprime. By (3.9) and induction, any integer that divides a and b divides every element that follows a and b on the same row; but the last element of the row is 1. As a side effect, if $C(a,b,c)$ holds and b is at an odd location (so that $b=a+c$) then a and c are coprime.

Our proof is as follows: if $C(a,b,c)$ holds, then $\gcd(a,b)=\gcd(b,c)$. So, the gcd of two consecutive elements does no depend on the position in the row. If we consider the first two elements of the row we get: the gcd of any two consecutive elements $S(u,v)$ is $\gcd(u,v)$.

```
Lemma sternD_gcd_rec u v i n  (S := sternD u v n):
  0 < i < 2^n ->
  gcdn (nth 0 S i.-1) (nth 0 S i) = gcdn (nth 0 S i)(nth 0 S i.+1).
Lemma sternD_gcd_rec u v i n  (S := sternD u v n):
   i < 2^n ->
  gcdn (nth 0 S 0) (nth 0 S 1) = gcdn (nth 0 S i)(nth 0 S i.+1).
Lemma sternD_gcd_rec2 u v i n  (S := sternD u v n):
   i < 2^n -> gcdn (nth 0 S i)(nth 0 S i.+1) = gcdn u v.
Lemma sternD_consecutive_coprime1 i n  (S := stern11 n):
   i < 2^n -> coprime (nth 0 S i)(nth 0 S i.+1).
Lemma sternD_coprime_succ_add i n  (S := stern11 n)
  (a := nth 0 S i.-1)(b := nth 0 S i)(c := nth 0 S i.+1):
  0 < i < 2 ^n -> b = a + c -> coprime a c.
```

In §6 Stern explains that $C(a,b)$ occurs at most once. His argument is a bit strange, and not really convincing. Assume first $a>b$. In this case, a is at an odd position, is not the first element of the row, thus is preceded by β, and $\beta+b=a$. On the previous row we have

$\mathbf{B}(\beta, b)$. The result holds by induction on the index of the row; however Stern considers the two cases $\beta < b$ and $\beta > b$ and the row preceding these two quantities, and recursively constructs (a_j, b_j) where $C(a_j, b_j)$ holds, and something (it is unclear what) is decreasing, so that finally one of a_j or b_j should be equal to 1. Now it is obvious that 1 can be followed (or preceded) by some number c at a unique position (the start or end of row $c-1$). Hence, the value b_j such that $a_j = 1$ (or the value a_j such that $b_j = 1$), together with j uniquely determines the row containing a and b. The trouble is, if a and b occur somewhere else, and if (a'_j, b'_j) is the sequence associated to this position, it is not clear why it should be equal to (a_i, b_i). Stern says that the case $b < a$ is similar. The cases $a = b$ and $\beta = b$ are not considered (two consecutive elements are coprime; so if they are equal, they are equal to 1, the only possibility being that they are on row number zero). So Stern concludes: if a and b are consecutive on rows k and k', then $k = k'$. Moreover, by a similar argument, a and b cannot be consecutive more than once on the same row.

[Recall that $C(a, b)$ means that for some k and i, where $i < 2^k$ we have $a = \mathbf{B}_k(i)$ and $b = \mathbf{B}_k(i+1)$; since $a = s(2^k + i)$ and $b = s(2^k + i + 1)$, Stern proves here that $j \mapsto (s(j), s(j+1))$ is injective. See proof in the previous chapter.]

```
Lemma Stern_pair_injective i n j m:
   i < 2^n -> j < 2^m ->
  nth 0 (stern11 n) i = nth 0 (stern11 m) j ->
  nth 0 (stern11 n) i.+1 = nth 0 (stern11 m) j.+1 ->
  i = j /\ n = m.
```

Consequences: if (Ca, b) holds on some row, then $C(b, a)$ holds on the same row (by symmetry), so cannot hold on another row. Moreover, if $C(a, b)$ holds on the first half, then $C(b, a)$ holds on the second.

In §8 Stern proves that $a \perp b$ implies $C(a, b)$. As previously one may assume $a < b$ (here again, Stern does not consider the case $a = b$; since a and b are coprime we have $a = b = 1$ and this pair appears on row zero). It is clear by induction on q that $C(a, r)$ implies $C(a, aq + r)$. As $C(1, 1)$ holds, so holds $C(1, b)$ whenever $b > 0$. So one may assume $a > 1$. Now consider $b = aq + r$. If $r = 0$, then $a = \gcd(a, b) = 1$, absurd. So $0 < r < a$. By induction $C(r, a)$ holds, thus the result.

```
Definition Stern_consecutive a b:=
  exists n i,
    [/\ i < 2^n, nth 0 (stern11 n) i = a  & nth 0 (stern11 n) i.+1 = b].

Lemma Stern_consecutive_sym a b:
   Stern_consecutive a b -> Stern_consecutive b a.
Lemma Stern_consecutive_nn a : Stern_consecutive a a ->  a = 1.
Lemma Stern_consecutive_rec a r n: Stern_consecutive a r ->
   Stern_consecutive a (n *a + r).
Lemma Stern_consecutive_1n n: Stern_consecutive 1 n.+1.
Lemma Stern_consecutive_coprime2 a b : 0 < a -> 0 < b -> coprime a b ->
   Stern_consecutive a b.
```

In §7 Stern states: the number of times an integer b appears in the table at an odd location is at most $\phi(b)$, and in §8 he shows that this is equal to $\phi(b)$. To fix ideas, we assume that b appears on row k, column $2i + 1$. Then $b = s_m$ where $m = 2^k + 2i + 1$; if $n = 2^{k-1} + i$ we get $b = s_{2n+1}$. The case $b = 1$ is a bit special, it will be excluded (we know that 1 appears as first

and last element on each row, the only case where it appears at odd position is $k=0$, $i=0$, this gives $m=2$, however there is exactly one n, namely zero, such that $1=s_{2n+1}$). Let

$$\mathrm{E}=\{a,\,0<a<b,a\perp b\},\quad \mathrm{F}=\{n,\,b=s_{2n+1}\},\quad \mathrm{G}=\{(k,i),\,2i+1<2^k,\mathrm{S}_k(2i+1)=b\}.$$

By the preceding remark, the two sets F and G have the same cardinal. As $b\neq 1$, the condition $0<a$ in E becomes useless so that E has cardinal $\phi(n)$. Take (k,i) in G. Let $a=\mathbf{B}_k(2i)=\mathbf{B}_{k-1}(i)$ and $c=\mathbf{B}_k(2i+2)=\mathbf{B}_{k-1}(i+1)$. Then $b=a+c$, so that $a\in\mathrm{E}$. This gives a function $\mathrm{G}\to\mathrm{E}$. It is surjective, since if $a\in\mathrm{E}$ and $c=b-a$, then a and c are coprime, so that is (k,i) such that $a=\mathbf{B}_k(i)$ and $c=\mathbf{B}_k(i+1)$, and $b=\mathbf{B}_{k+1}(2+1)$. Since there is a unique location where a precedes b, this function is injective.

In §9, Stern states that row $b-1$ is the last in which b occurs on odd position (the value of an element at odd position on row k is $>k$). In terms of the fusc function, this is $n<2^{s_{2n+1}-1}$; so that $n\in\mathrm{F}$ implies $n<2^{b-1}$. It follows that

$$\mathrm{card}(\mathrm{F})=\sum_{i<2^{b-1}}[b=s_{2i+1}]$$

where [B] is zero if B is false and one if B is true. We show here that this sum is $\phi(b)$, with the arguments exposed above.

```
Lemma sternD_location_bound b:
   let H := fun k => exists i, b = nth 0 (stern11 k) i.*2.+1 in
   0 < b -> H b.-1 /\  forall k, H k -> k < b.
Lemma fusc_totient b: 0 < b ->
  \sum_(i< 2^b.-1) (b== fusc (i.*2.+1)) = totient b.
```

Stern concludes §9 with: the number of occurrences of b on row k is $\phi(b)$ whenever $k<b$. In particular b is prime if and only if the number of occurrences on row $b-1$ is $b-1$. In terms of the fusc function, the statement becomes: the number of indices i such that $2^{b-1}\le i<b$ and $s_i=b$ is $\phi(b)$ (write $i=(2q+1)2^k$, then $b_i=b_{2q+1}$ and the two quantities q and k are uniquely determined by i).

```
Lemma fusc_totient2 n b: 0 < b -> b <= n.+1 ->
  \sum_(2^n <= i< 2^(n.+1)) (b== fusc i) = totient b.
Lemma fusc_totient_prime b
  (H := fun b =>  \sum_(2^(b.-1) <= i< 2^ b) (b== fusc i) = b .-1) :
  (prime b -> H b) /\ (0 < b -> H b  -> prime b).
```

Stern concludes §11 as follows. Let T be a list of numbers, and $q(\mathrm{T})$ be the function $i\mapsto(\mathrm{T}_i+\mathrm{T}_{i+2})/\mathrm{T}_{i+1}$. He pretends that the three functions $q(\mathbf{B}_{n+1})$, $q(\mathbf{B}_n)$, and $q(\mathbf{A}_n)$ are equal. (in §4 he proved that $q(\mathbf{B}_n)$ is independent of n and concludes by $\mathbf{B}_{n+1}(i)=\mathbf{B}_n(i)+\mathbf{A}_n(i)$). As mentioned above, if T is any $\mathrm{S}_n(u,v)$ the quantity $q(\mathrm{T})(i)$ depends only on i.

In §12 Stern proves

$$\mathbf{B}_n(i)\mathbf{B}_{n+1}(i+1)-\mathbf{B}_{n+1}(i)\mathbf{B}_n(i+1)=1,$$

$$\mathbf{B}_n(i)\mathbf{A}_n(i+1)-\mathbf{A}_n(i)\mathbf{B}_n(i+1)=1,$$

and claims that the second relation is the best Bezout relation between $\mathbf{B}_n(i)$ and $\mathbf{B}_n(i+1)$. The two relations are clearly equivalent as $\mathbf{B}_{n+1}=\mathbf{B}_n+\mathbf{A}_n$. Moreover, $\mathbf{A}_n(j)\le\mathbf{B}_n(j)$ whatever j, and equality holds only when $j=2^n$. This ensures minimality.

For simplicity, we shall use equation (3.6) from the previous chapter, restated as

$$s_{2^n+i}s_{i+1}=1+s_{2^n+i+1}s_i. \tag{4.7}$$

```
Lemma sternD_bezout n i (f := fun n i => nth 0 (stern11 n) i):
    i < 2 ^n -> (f n i) * (f n.+1 i.+1) = 1 +  (f n i.+1) * (f n.+1 i).
Lemma fusc_bezout_minimal n i
    (a := fusc (2 ^ n + i))(b:= fusc (2 ^ n + i.+1)):
   i < 2 ^n ->
   fusc i.+1 * a = fusc i * b + 1
   /\ (forall u v, u* a = v * b +1 -> (fusc i.+1 <= u) && (fusc i <= v)).
```

4.3 The case S(1,n)

In §13 and following, Stern considers $S(1,n)$ where $n > 1$. Let x be the element at row i, column q. By linearity $x = k + ln$, for some integers k and l (namely $k = s(2^q - i)$ and $l = s(i)$). We have seen in the previous chapter that these quantities are coprime (see also below). On the other hand, the q-th row of $S(1,n)$ is the start of the $q+n-1$-th row of $S(1,1)$. So $S_q(1,n+1)(i) = S_{q+n}(1,1)(i) = s(2^{q+n} + i)$.

```
Lemma sternD_1n_kl n q i: i <= 2^q ->
   nth 0 (sternD 1 n q) i = stern (2 ^ q - i) + n * stern i.
Lemma sternD_1n_kl_sym n q i j (k :=  stern j) (l := stern i):
  i + j = 2^q ->  nth 0 (sternD 1 n q) i = k + n * l /\ coprime k l.
Lemma sternD_1n_v n q i:
  i <= 2^q -> nth 0 (sternD 1 n.+1 q) i = nth 0 (stern11 (q + n)) i.
Lemma sternD_1n_v' n q i:
  i <= 2^q -> nth 0 (sternD 1 n.+1 q) i = stern (2^ (q + n) + i).
```

Most properties of $S(1,1)$ hold; for instance equation (4.6); and that if a and c are consecutive somewhere, there are coprime and consecutive at a unique location. However, at least half of the table is lost. One can express this as: at least one of $C(a,b)$ and $C(b,a)$ is false. If fact, if $C(a,b)$ (or $C(b,a)$) holds on row k for $S(1,n)$ it holds on the first half of row $k+n-1$ for $S(1,1)$. But, at the end of §6, Stern said that one of $C(a,b)$ or $C(b,a)$ must occur on the second half.

Stern notices that $S(1,n)$ contains 1 and all integers $\geq n$, but nothing else (note that row k, column 1 holds $n+k$; the result is true even for $n = 0$).

```
Lemma sternD_1n_small n (S := sternD 1 n.+2)
   (C := fun a b => exists i k,
    [/\ i < 2^k, a = nth 0 (S k) i & b = nth 0 (S k) i.+1]) a b:
  C a b -> C b a -> False.
Lemma sternD_1n_member a n:
  (exists k i, i <=2^k /\ a =  nth 0 (sternD 1 n k) i) <->
  (a==1 \/ n <= a).
```

Stern says that $S(1,n)$ is absolutely not symmetric. In other terms, if y is the symmetric of x on the same row, then $x \neq y$. Note that $y = S_q(1,n)(2^q - i)$ so that $y = l + kn$. The result comes from: in the first half of the row $k > l$, in the second half $k < l$, and in the middle $k = l = 1$. The result seems obvious to Stern. It is equivalent to: if $i < j$ and $i + j = 2^q$ then $s_i < s_j$, which has been shown in the previous chapter (by induction).

We first consider a property that Stern states in the next section (§14). First, in the first half of row $q+1$ of $S(1,n)$, k and l are respectively the elements of row q of $S(1,1)$ and $S(0,1)$; in the second half they are the elements of $S(1,0)$ and $S(1,1)$. This means $S_{q+1}(1,n)(i+2^q) = S_q(1,0)(i) + n \cdot S_q(1,1)(i)$. It is however easier to start counting from the right. We have then $S_{q+1}(1,n)(2^{q+1} - i) = S_q(0,1)(i) + n \cdot S_q(1,1)(i)$.

Assume now that x and y are symmetric, on row $q+1$, x is on the left, at position i. Now $x = k + ln$, where $k = \mathrm{S}_q(1,1)(i)$, $l = \mathrm{S}_q(0,1)(i)$. Since $k = s(2^q - i) + l$, we get $l < k$, thus $x < y$.

```
Lemma sternD_1n_H1 q n i: i <= 2^q ->
   nth 0 (sternD 1 n q.+1) i = nth 0 (stern11 q) i + n * nth 0 (stern01 q) i.
Lemma sternD_1n_H2 q n i: i <= 2^ q ->
   nth 0 (sternD 1 n q.+1) (i + 2^q) =
   nth 0 (sternD 1 0 q) i + n * nth 0 (stern11 q) i.
Lemma sternD_1n_H2' q n i: i <= 2^ q ->
   nth 0 (sternD 1 n q.+1) (2^q.+1 - i) =
   nth 0 (stern01 q) i + n * nth 0 (stern11 q) i.
Lemma sternD_1n_disymmetric n q i j (S := sternD 1 n.+2 q) :
   i < j -> i + j = 2^q -> nth 0 S i < nth 0 S j.
```

In §14 Stern says: consider two consecutive positions on some row with values $k + ln$ and $k' + l'n$. Then $kl' - k'l = 1$, and this is the best Bezout relation for k and k'. The relation follows from (4.7) on the first half, and holds on the second half by symmetry. This is however not the best relation: we have $k = s(j)$ and $l = s(2^q - j)$ where j is the position on the row, counted from the right. If j is fixed and q becomes large, then l and l' become large, thus cannot be minimal (for instance, when $j = 1$, we get $k = 1$ and $k' = 0$, $l = q$ and $l' = 0$).

Stern says: the table contains no element of the form $hk + h'kn$. This is strange.

```
Lemma sternD_1n_bezout n q i (S := sternD 1 n q)
  (k:= stern (2 ^ q - i)) (l:= stern i)
  (k':=  stern (2 ^ q - i.+1)) (l':= stern i.+1):
  i < 2^q ->
  [/\ nth 0 S i = k + n*l, nth 0 S i.+1 = k' + n*l' &  k * l' =  k' * l + 1].
```

In §15 Stern says: the elements of $\mathrm{S}(1,n)$ at odd position are those of the form $\mathrm{K} + \mathrm{L}n$, where $\mathrm{K} \perp \mathrm{L}$, and this form is unique (recall that elements at even position are copies of elements at odd position). Existence: Stern says that, by symmetry, he may assume $\mathrm{K} > \mathrm{L}$ (as usual, he forgets the case $\mathrm{K} = \mathrm{L}$). He considers the least solution of $\mathrm{K}x - \mathrm{L}y = 1$, say $x = \mathrm{L}_0$, and $y = \mathrm{K}_0$ ($x \le \mathrm{L}$ and $y \le \mathrm{K}$). The minimal solution of $\mathrm{K}x - \mathrm{K}_0 y = 1$ is $x = \mathrm{L}_0$ and $y = \mathrm{L}$ (since $\mathrm{L} < \mathrm{K}$). This means that there is a pair of consecutive numbers somewhere, of the form $k + ln$, $k' + l'n$, with $k = \mathrm{K}$ and $k' = \mathrm{K}_0$. By minimality, it follows $\mathrm{L} = l$ and $l' = \mathrm{L}_0$. In particular, there is a number with value $\mathrm{K} + \mathrm{L}n$ (Note: we mentioned above that $kl' - k'l = 1$ is not always the best Bezout relation; it seems however to be true on the first part of the row, which means $\mathrm{K}_0 < \mathrm{K}$, so maybe the reasoning is correct).

This needs to be completed.

```
Lemma sternD_1n_coprime n q i: i < 2^q -> exists K L,
  coprime K L /\ nth 0 (sternD 1 n q) i = K + n*L.
```

There are other sections in Stern's paper. They are not yet implemented in Coq, but will be in a near future. In particular, Stern considers continued fractions associated to k/l and the Bezout relation $kl' - k'l = 1$. This is somehow relation to the Stern-Brocot tree (see end of the previous chapter).

Bibliography

[1] Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development*. Springer, 2004.

[2] J.L. Brown, Jr. A new characterization of the Fibonacci numbers. *Fibonacci Quaterly*, 3(1), 1965.

[3] Georg Cantor. *Contributions to the Founding of the Theory of Transfinite Numbers*. Dover Publications Inc, 1897. Trans. P. Jourdain, 1955.

[4] L. Carlitz. A note on Fibonacci numbers. *Fibonacci Quaterly*, 2(1), 1964.

[5] L. Carlitz. Fibonacci representations. *Fibonacci Quaterly*, 1968.

[6] Edsgert Dijkstra. EWD578: More about the function "fusc" (a sequel to EWD570). http://www.cs.utexas.edu/users/EWD/ewd05xx/EWD578.PDF, 1976. In "Selected Writings on Computing: A Personal Perspective, Springer-Verlag, 1982. ISBN 0–387–90652–5".

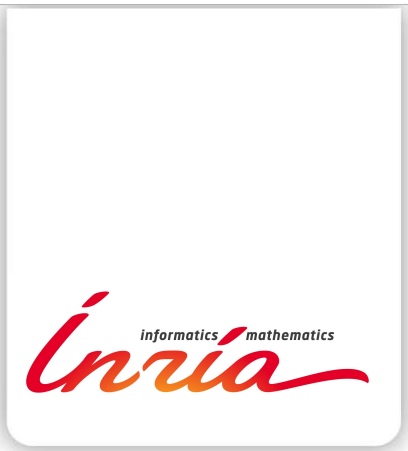
[7] José Grimm. Implementation of Bourbaki's Elements of Mathematics in Coq: Part Two; Ordered Sets, Cardinals, Integers. Research Report RR-7150, INRIA, 2009. http://hal.inria.fr/inria-00440786/en/.

[8] Milad Niqui. Exact arithmetic on the Stern-Brocot tree. *Journal of Discrete Algorithms*, 5(2), 2007.

[9] Milad Niqui and Yves Bertot. Qarith: Coq formalisation of lazy rational arithmetic. In Stefano Berardi, Mario Coppo, and Ferruccio Damiani, editors, *Types for Proofs and Programs*, volume 3085 of *Lecture Notes in Computer Science*, pages 309–323. Springer Berlin Heidelberg, 2004.

[10] Moritz Stern. Ueber eine zahlentheorische Funktion. *Journal für die reine und angewandte Mathematik*, 55:193–220, 1858.

Contents



RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399